Normal Factor Graphs as Probabilistic Models

Ali Al-Bashabsheh and Yongyi Mao
School of Electrical Engineering and Computer Science
University of Ottawa, Canada
{aalba059, yymao}@site.uottawa.ca

Abstract

We present a new probabilistic modelling framework based on the recent notion of normal factor graph (NFG). We show that the proposed NFG models and their transformations unify some existing models such as factor graphs, convolutional factor graphs, and cumulative distribution networks. The two subclasses of the NFG models, namely the constrained and generative models, exhibit a duality in their dependence structure. Transformation of NFG models further extends the power of this modelling framework. We point out the well-known NFG representations of parity and generator realizations of a linear code as generative and constrained models, and comment on a more prevailing duality in this context. Finally, we address the algorithmic aspect of computing the exterior function of NFGs and the inference problem on NFGs.

1 Introduction

In the recent years, probabilistic graphical models have emerged from different disciplines as a powerful methodology for statistical inference and machine learning. Traditional such models, such as Bayesian networks [1] and Markov random fields [2], primarily aim at representing the joint probability distribution (i.e., probability mass function or probability density function) of the random variables (RVs) of interest in terms of their multiplicative factorization structure. Such "multiplicative" modelling semantics can be translated to the language of factor graphs (FGs) [3], a mathematical and graphical framework that is convenient and intuitive for representing the multiplicative factorization of a multivariate function. It is arguable that FGs and their variants, such as directed factor graphs [4], unify the various such multiplicative models [3, 4]. In contrast to the multiplicative models, convolutional factor graphs (CFGs) [5, 6] are models which represent the joint distribution of interest in terms of convolutional factorizations. The CFG modelling framework has recently demonstrated its power in a derivative of the CFG model, known as linear characteristic models (LCM) [7], for inference in stable distributions. Instead of directly representing the distribution of interest, LCM represents the characteristic function of the distribution. This is advantageous for stable distributions, which are only explicitly defined in the characteristic function domain. We argue that the philosophy of modelling in a "transform domain", as manifested in LCM, should not be overlooked. This is because the unique nature of an inference task one is faced with may favour a representation of some other objects than the probability distribution. Incidentally or not, cumulative distribution functions, which may be viewed as transformations of probability distributions, appear more favourable in structured ranking problems, and this recognition has led to the development of cumulative distribution networks (CDNs) [8].

In this paper, we present a new graphical model, the normal factor graph model, based on the notion of normal factor graphs (NFGs) [9, 10]. In the framework of NFGs, a powerful tool, called holographic transformation, has been developed . It was shown in [9] that this tool unifies a duality theorem of Forney [11] in coding theory and the Holant theorem of Valiant [12] in complexity theory.

The main objective of this paper is to show that the proposed NFG models, together with the holographic transformation technique, essentially unifies all the probabilistic models mentioned above. We will focus on two subfamilies of NFG models, namely constrained and generative NFG models. We will show that constrained NFG models reduce to FGs, however, have a different interpretation, and that generative NFG models, restricted to a special case, reduce to CFGs. In addition, we reveal an interesting "duality" between the constrained and generative NFG models in their independence properties. A general model transformation technique is introduced, using which we show a CDN is equivalent to a transformed NFG model.

2 Probabilistic Graphical Models

Here we give a brief summary of the previous graphical models relevant to this work. As a notational convention that will be used throughout the paper, a RV is denoted by a capitalized letter, for example, by X, $Y, \ldots$, and the value it takes will be denoted by the corresponding lower-cased letter, i.e., $x, y, \ldots$.

2.1 Factor Graphs

A *factor graph* (FG) [3] is a bipartite graph $(V \cup U, E)$ with independent vertex sets V and U, and edge set E, where each vertex $v \in V$ is associated a variable x_v from a finite alphabet $\mathcal{X}_v$, and each vertex $u \in U$ is associated a complex-valued function f_u on the cartesian product $\mathcal{X}_{\mathrm{ne}(u)} := \prod\limits_{v \in \mathrm{ne}(u)} \mathcal{X}_v$, where $\mathrm{ne}(u) := \{v \in V : \{u, v\} \in E\}$ is the set of neighbors (adjacent vertices) of u. Each function f_u is referred to as a *local* function and the FG is said to *represent* a function given by $f(x_V) := \prod_{u \in U} f_u(x_{\mathrm{ne}(u)})$, where we use the "variable set" notation, defined for any $A \subseteq V$, as $x_A := \{x_a : a \in A\}$. In the context of FGs, the function represented by the FG is often called the *global function*. Fig. 1 (a) is an example FG.

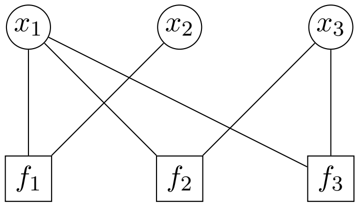


Figure 1: An example of an FG, a CFG, and a CDN: (a) When viewed as an FG, the graph represents the global function $f_1(x_1, x_2)f_2(x_1, x_3)f_3(x_1, x_3)$, (b) as a CFG, the graph represents the global function $f_1(x_1, x_2) * f_2(x_1, x_3) * f_3(x_1, x_3)$, and (c) as a CDN, the graph is understood as an FG where each local function is a cumulative distribution, in which case, the global function $f_1(x_1, x_2)f_2(x_1, x_3)f_3(x_1, x_3)$ satisfies the properties of a cumulative function, and is taken as the joint cumulative distribution of the RVs X_1, X_2 and X_3.

Since independence (or conditional independence) relationships among RVs are often captured via the multiplicative factorization of their joint probability distribution, FGs, when used to represent the joint distribution of RVs, form a convenient probabilistic model.

The relationship between FG probabilistic model and other classical probabilistic models, such as Bayesian networks and Markov random fields, is well-known, see, e.g. [3]. In these models, all featuring the "multiplicative semantics" and aiming at representing the joint distributions, efficient inference algorithms, such as the belief propagation or the sum-product algorithm, have been developed and demonstrated great power in various applications.

2.2 Convolutional Factor Graphs

Let $\mathcal{X}_1, \mathcal{X}_2$ and $\mathcal{X}_3$ be three (possibly distinct) finite sets. In general, we require the sets to have an "abelian group" structure, so that a notion of addition "$+$" and its inverse "$-$" are well defined. The requirement that the sets be finite is not particularly critical but only for the convenience of argument.

Let f_1 and f_2 be two function on $\mathcal{X}_1 \times \mathcal{X}_2$ and $\mathcal{X}_2 \times \mathcal{X}_3$, respectively. The *convolution* of f_1 and f_2, denoted $f_1 * f_2$, is defined as the function on $\mathcal{X}_1 \times \mathcal{X}_2 \times \mathcal{X}_3$ given by, $(f_1 * f_2)(x_1, x_2, x_3) := \sum_{x \in \mathcal{X}_2} f_1(x_1, x_2 - x) f_2(x, x_3)$. Following the convention in [5], we may write $(f_1 * f_2)(x_1, x_2, x_3)$ as $f_1(x_1, x_2) * f_2(x_2, x_3)$ to emphasize the domains of the original functions. It is not hard to show that the convolution as defined above is both associative and commutative.

A *convolutional factor graph* (CFG) [6] is a bipartite graph that represents a global function that factors as the convolution of local functions. In fact, the representation semantics in a CFG is identical to that in an FG (often referred to as a "multiplicative" FG for distinction), except that the above defined notion of convolution is used as the product operation, cf. Fig. 1 (b) for an example CFG. In [5] CFGs were presented as a probabilistic graphical model to represent the joint probability distribution of a set of observed RVs that are constructed from a collection of independent sets of latent RVs via linear combinations. In addition, the authors of [6] presented an elegant duality result between FGs and CFGs via the Fourier transform. Such a duality and CFGs have recently been exploited by [7] in what is known as linear characteristic model (LCM) for solving inference problems with stable distributions.

2.3 Cumulative Distribution Networks

Let X be a RV assuming its values from a finite ordered set $\mathcal{X}$. The *cumulative distribution function* (CDF) of X is defined as $F_X(x) := \sum_{y \leq x} p_X(y)$, where p_X is the probability distribution of X. We note that this definition of CDF, as a function on $\mathcal{X}$, is slightly different from the classical definition of CDF, which is a function defined on the real line (or on the Euclidean space in the multivariate case). It nevertheless captures the same essence and is merely a different representation, suitable and convenient in the context of this paper. Such a notion of CDF can be extended to any collection of RVs $X_1, \ldots, X_n$ assuming their values from the finite ordered sets $\mathcal{X}_1, \ldots, \mathcal{X}_n$ by defining their joint CDF as $F_{X_1,\ldots,X_n}(x_1, \ldots, x_n) := \sum_{y_1 \leq x_1, \ldots, y_n \leq x_n} p_{X_1,\ldots,X_n}(y_1, \ldots, y_n)$, where $p_{X_1,\ldots,X_n}$ is the joint probability distribution of $X_1, \ldots, X_n$. Note that while the marginal probability distribution is computed by *summing* the joint probability distribution over the range of the marginalized RVs, the marginal CDF is computed by *evaluating* the joint CDF at the largest element of $\mathcal{X}_i$, for all marginalized RVs X_i. (That is, if I indexes the set of marginalized RVs and X_i takes its values from the ordered set $\{1, \ldots, |\mathcal{X}_i|\}$, then we evaluate the joint CDF at $|\mathcal{X}_i|$ for all $i \in I$.) It is well known that CDFs satisfy a collection of properties as were articulated in standard textbooks and in [8]. On the other hand, any function satisfying such properties, which we shall refer to as "CDF axioms", may be regarded as a CDF and can be used to define a collection of RVs.

A *cumulative distribution network* (CDN) [8] is a multiplicative FG in which each local function satisfies the CDF axioms, then it is straightforward to show that the global function represented by the FG also satisfies the CDF axioms. The global function thus defines a collection of random variables, each represented by a variable node in the FG, and the CDN may serve as a probabilistic model. In [8], it was shown that CDNs are useful for structured ranking problems, and efficient inference algorithms for such problems were developed in these models. See Fig. 1 (c) for an example CDN.

3 Normal Factor Graphs

Now we give a quick overview of the framework of normal factor graphs (NFGs), and develop some notations and definitions for subsequent discussions.

3.1 NFG and the exterior function

A *normal factor graph* (NFG) [9, 10] is a graph (V, E), with vertex set V and edge set E, where the edge set E consists of two types of edges, a set T of regular edges (also called internal edges), each incident on two vertices, and a set L of "half edges" (also called external edges or dangling edges), each incident on exactly one vertex. Every edge $e \in E$ is associated a variable x_e from a finite alphabet $\mathcal{X}_e$, and every vertex $v \in V$ is associated a *local* function f_v on the cartesian-product $\mathcal{X}_{E(v)} := \prod\limits_{e \in E(v)} \mathcal{X}_e$, where $E(v)$ is the set of (internal and external) edges incident on v. At some places we may use $T(v)$ and $L(v)$ to denote the internal and external edges incident on vertex v, respectively, and it is clear that $E(v) = T(v) \cup L(v)$ for all v. We use the symbol $\mathcal{G}$ to refer to an NFG, and sometimes write $\mathcal{G}(V, E, f_V)$, where $f_V := \{f_v : v \in V\}$, to emphasize the NFG parameters.

An NFG $\mathcal{G}$ is associated with a function, called the *exterior function* and denoted by $Z_{\mathcal{G}}$, on the cartesian product $\mathcal{X}_L := \prod\limits_{e \in L} \mathcal{X}_e$, defined as

$$Z_{\mathcal{G}}(x_L) := \sum_{x_T} \prod_{v \in V} f_v(x_{E(v)}).$$

That is, the exterior function realized by an NFG is the product of all its local functions with the internal variables (edges) summed over. An example NFG is shown in Fig. 2.

Let $V = \{1, \ldots, |V|\}$, at some places for notational convenience, we may denote the right hand side of the above equation, a "sum-of-products form," by $\langle f_1, f_2, \ldots, f_{|V|} \rangle$. This notation is valid due to the distribution law, and the associativity and commutativity of addition and multiplication, making the bracketing and ordering of the arguments in the notation $\langle \cdot, \cdots, \cdot \rangle$ irrelevant. For example, the sum-of-products form encoded by the NFG in Figure 2 may be written as $\langle f_1(x_1, s_1, s_2), f_2(x_2, s_2, s_3, s_5), f_3(s_3, s_4), f_4(s_1, s_4, s_5) \rangle$, or even $\langle f_1, f_2, f_3, f_4 \rangle$ for simplicity.

We say that two NFGs are *equivalent* if they realize the same exterior function. At some places, we may extend this notion of equivalence to include other graphical models and say, for instance,

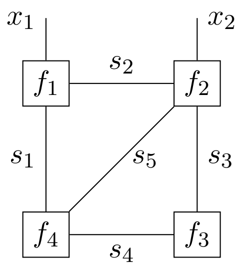


Figure 2: An NFG realizing the exterior function $\sum_{s_1,\ldots,s_5} f_1(x_1, s_1, s_2)f_2(x_2, s_2, s_3, s_5)f_3(s_3, s_4)f_4(s_1, s_4, s_5)$.

"an FG is equivalent to an NFG," where we mean that the product function of the FG is equal to the exterior function of the NFG.

Finally, we call an NFG with no loops or parallel internal edges a *simple* NFG, and an NFG with a bipartite underlying graph $(I \cup J, E)$ a *bipartite* NFG, where I and J are the two independent vertex sets. We impose no restriction on the cycle structure of NFGs.

3.2 Special kinds of local functions

The following (non-disjoint) classes of local functions will be of particular interest.

Split functions Let $\mathcal{X}_1, \ldots, \mathcal{X}_n$ be some finite sets, then we say a function f on $\mathcal{X}_1 \times \cdots \times \mathcal{X}_n$ is a *split* function via x_1, and refer to x_1 as the *splitting variable*, if

$$f(x_1, \ldots, x_n) = f_2(x_1, x_2)f_3(x_1, x_3) \ldots f_n(x_1, x_n),$$

for some bivariate functions $f_2, \ldots, f_n$. Note that it follows immediately that any bivariate function is trivially a split function (via any of its arguments). Subsequently, if we do not explicitly specify the splitting variable of a split function, then it is assumed to be the function's first argument. Graphically, we draw a split function as in Fig. 3 (a), where the in-ward directed edge is used to distinguish the splitting argument, and the remaining arguments are successively encountered in a counter clock-wise manner with respect to the directed edge.

Conditional functions Let $\mathcal{X}_1, \ldots, \mathcal{X}_n$ be some finite sets, then a function f on $\mathcal{X}_1 \times \cdots \times \mathcal{X}_n$ is said to be a *conditional* function of x_1 given $x_2, \ldots, x_n$ if there is a constant c such that $\sum_{x_1} f(x_1, \ldots, x_n) = c$, for all $x_2, \ldots, x_n$. It is apparent that a non-negative real conditional function with $c = 1$ is a conditional probability distribution. A conditional function is shown in Fig. 3 (b), where we use the same convention of edge labeling as in the case of split functions, but with an out-ward directed edge to mark the first argument.

One may observe a sense of "duality" between a conditional function and a split function through the following lemma.

Lemma 1. *Let $f(x_1, x_2, x_3)$ be a positive real split function, then up to a scaling factor, f may be written as $p_{X_1}(x_1)p_{X_2|X_1}(x_2|x_1)p_{X_3|X_1}(x_3|x_1)$ for some probability distributions p_{X_1}, $p_{X_2|X_1}$, and $p_{X_3|X_1}$.*

Proof. Since f is a positive real function, it may be viewed (up to a scaling factor) as a probability distribution of some RVs X_1, X_2 and X_3. Hence, f can be written as $f(x_1, x_2, x_3) = p_{X_1}(x_1)p_{X_2|X_1}(x_2|x_1)p_{X_3|X_1X_2}(x_3|x_1, x_2)$. Since f is a split function via x_1, as we will see later (cf. Lemma 4) , we have $X_2 \perp\!\!\!\perp X_3|X_1$, and the claim follows. $\square$

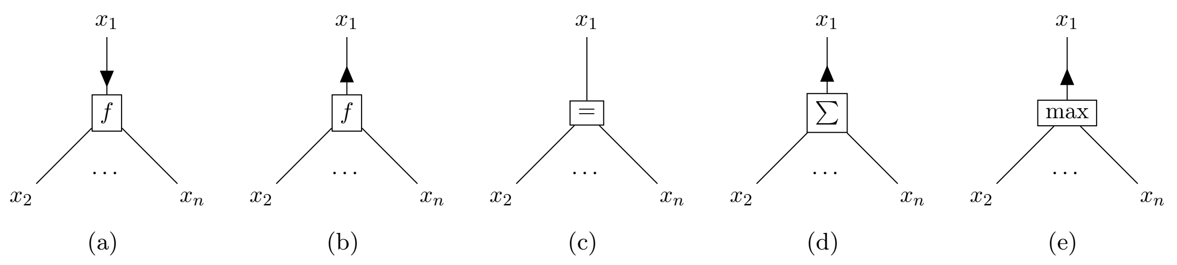


Figure 3: A graphical illustration of: (a) split function (b) conditional function (c) $\delta_=$, (d) δ_Σ, and (e) $\delta_{\max}$.

Now compare a split function $f(x_1, x_2, x_3)$ with a conditional function $g(x_1, x_2, x_3)$ where let us assume that the respective scaling constants making the functions into distributions are both 1. If we are to draw the Bayesian networks (BN) [1] corresponding to the two distributions f and g respectively, we shall see that the directions of the edges in the BN of f are completely opposite to those in the BN of g. Describing it in terms of causality, one may say: The distribution f prescribes that conditioned on the RV X_1, we generate the RVs X_2 and X_3 *independently*, whereas the distribution g prescribes that X_2 and X_3 generates X_1 *jointly*. This sense of "duality" or "reciprocity" (evidently existing in the two kinds of functions involving arbitrary number of variables) also justifies our notations of opposite edge directions in denoting the two kinds of functions.

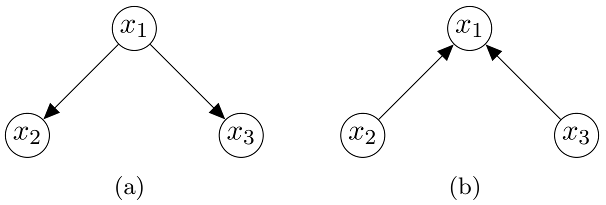


Figure 4: The BNs corresponding to: (a) a split function, and (b) a conditional function.

Indicator functions and the Iverson's convention An *indicator function* is a $\{0,1\}$-valued function, and at many places we will use the Iverson's convention $[P]$ to denote the indicator function on the true/false proposition P defined as $[P] := 1$ if and only if P is true. For any $x, y \in \mathcal{X}$, we will often use the proposition $x \overset{?}{=} y$, which is defined to be "true" if $x = y$ and to be "false" otherwise. Subsequently, we will use the symbol "=" instead of "$\overset{?}{=}$", and assume the distinction from the usual use of "=" for assignment is clear from the context— Namely, any occurrence of "=" inside, and only inside, the Iverson brackets refers to "$\overset{?}{=}$". We will mainly be interested in the following indicator functions:

Evaluation indicator Let $\mathcal{X}$ be a finite alphabet, the *evaluation indicator* (evaluating at some $\overline{x} \in \mathcal{X}$) is an indicator function on $\mathcal{X}$ defined as $\delta_{\overline{x}}(x) := [x = \overline{x}]$ for all $x \in \mathcal{X}$,

Equality indicator Let $\mathcal{X}$ be a finite alphabet, the *equality indicator* on n variables is defined as $\delta_=(x_1, \ldots, x_n) := \prod_{i=2}^{n} [x_1 = x_i]$ for all $x_1, \ldots, x_n \in \mathcal{X}$. Note that an equality indicator is a split function via any of its arguments, hence, at many places, we may illustrate it graphically without a directed edge.

Constant-one indicator Let $\mathcal{X}$ be a finite alphabet, the *constant-one indicator* is a degenerate indicator function defined as $\mathbf{1}(x) := 1$ for all $x \in \mathcal{X}$.

Sum indicator Let $\mathcal{X}$ be a finite abelian group (additively written), the *sum indicator* on n variables is defined as $\delta_\Sigma(x_1, \ldots, x_n) := [x_1 = x_2 + \cdots + x_n]$ for all $x_1, \ldots, x_n \in \mathcal{X}$. A closely related indicator function, which is more popular in the factor graphs and normal graphs literature, is the *parity indicator* function, denoted δ_+, and defined as $\delta_+(x_1, \ldots, x_n) := [x_1 + \cdots + x_n = 0]$. It is clear that $\delta_\Sigma(x_1, \ldots, x_n) = \delta_+(x_1, -x_2, \ldots, -x_n) = \delta_+(-x_1, x_2, \ldots, x_n)$.

An elementary result concerning sum indicator function is the following lemma.

Lemma 2. *For any functions f on $\mathcal{X}_1 \times \mathcal{X}_2$ and g on $\mathcal{X}_2 \times \mathcal{X}_3$, where $\mathcal{X}_1$, $\mathcal{X}_2$ and $\mathcal{X}_3$ are abelian groups,*

$$\sum_{t,u} f(x,t) g(u,z) \delta_{\sum}(y,t,u) = f(x,y) * g(y,z),$$

where $\delta_{\sum}$ above is defined on $\mathcal{X}_2^3$.

Proof. Follows directly from the definition of the convolution. □

Max indicator Let $\mathcal{X}$ be an ordered finite set. The *max indicator* on n variable is defined as $\delta_{\max}(x_1, \ldots, x_n) := [x_1 = \max(x_2, \ldots, x_n)]$ for all $x_1, \ldots, x_n \in \mathcal{X}$. Let I be a finite set and let $\mathcal{X}_i$ be an ordered finite set for all $i \in I$. The definition of the max indicator is extended to the partially-ordered set $\mathcal{X}_I$ by defining $\delta_{\max}(x_I, \ldots, x'_I) := \prod_{i \in I} \delta_{\max}(x_i, \ldots, x'_i)$ for all $x_I, \ldots, x'_I \in \mathcal{X}_I$.

A graphical illustration of the above local functions is shown in Fig. 3. Note that the max and sum indicator functions are both conditional functions as illustrated by the directed edges in Figs. 3 (d) and (e). It is worth noting that the bivariate max indicator and bivariate sum indicator are both equivalent to the bivariate equality indicator, which is a split and a conditional function.

3.3 Vertex merging/splitting and holographic transformations

In the framework of NFGs, a pair of graphical procedures, known as the *vertex merging* and *vertex splitting* procedures are particularly useful. In a *vertex merging* procedure, two vertices representing functions f and g are replaced with a single vertex representing the function $\langle f, g \rangle$; conversely, in a *vertex splitting* procedure, a vertex representing a function that can be expressed in the sum-of-products form $\langle f, g \rangle$ is replaced with an NFG realizing the function $\langle f, g \rangle$. The two procedures are illustrated in Fig. 5, and are closely related to the concept of opening/closing the box [13, 14]. Note that when we put a dashed box around f and g we mean that they are replaced with the single function node $\langle f, g \rangle$, in other words, the last two pictures in Fig. 5 refer to the same NFG. It is easy to see that the exterior function of any NFG is invariant under the vertex merging/splitting procedure [9].

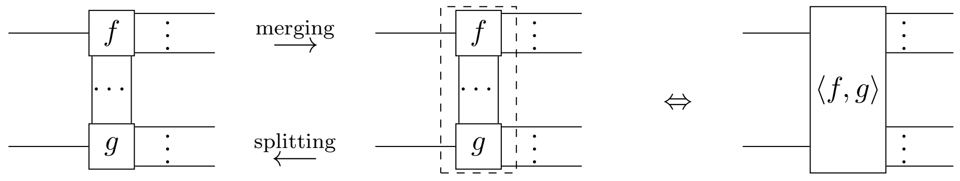


Figure 5: Vertex merging and vertex splitting.

The simple pair of procedures enables the notion of "holographic transformation" for NFGs, which transforms each local function of an NFG while keeping the graph topology. Further, the "generalized Holant theorem" relates the exterior function realized by the holographically transformed NFG with that realized by the original NFG. We summarize these notions below, and refer the interested reader to [9] for more details.

Suppose that f and g are two bivariate functions on $\mathcal{X}\times\mathcal{X}$ such that $\langle f(x,s), g(s,x')\rangle = \delta_=(x,x')$ for all $(x,x') \in \mathcal{X}\times\mathcal{X}$, then inserting to an NFG edge (regular or half), representing an $\mathcal{X}$-valued variable, the pair of functions f and g is equivalent to inserting the function $\delta_=$, which can be verified not to change the exterior function. The functions f and g in this case are called an inverse-pair of transformers, and such a graphical procedure is called *inverse-pair transformer insertion*.

Given an NFG $\mathcal{G}$ with a set of half edges L, the following procedure defines a transformed NFG with the same topology as $\mathcal{G}$:

(H1) In each half edge x_i, insert a bivariate function $g_i(x_i, y_i)$— We may refer to such transformers as *external transformers*.

(H2) In each internal edge, insert an inverse-pair of transformers— We may refer to such transformers as *internal transformers*.

(H3) For each original vertex v in $\mathcal{G}$, apply vertex merging procedure to merge f_v and its surrounding vertices.

Such a transformation of NFG is known as a *holographic transformation*. If we denote the resulting NFG by $\mathcal{G}'$, then it is clear that $Z_{\mathcal{G}'}(y_L) = \langle Z_{\mathcal{G}}(x_L), \prod_{i\in L} g_i(x_i,y_i)\rangle$, since only Step (H1) above affects the exterior function. This is essentially the *generalized Holant theorem* of [9], which in subsequent discussions will be referred to as the GHT.

4 NFG Models

We now present a generic NFG probabilistic model. Formally, an *NFG probabilistic model* or simply an *NFG model* is an NFG whose exterior function is up to scale the joint distribution of some RVs (each represented by a half edge) and which satisfies the following two properties: 1) the NFG is bipartite and simple; 2) half edges are only incident on one independent vertex set and there is exactly one half edge incident on each vertex in this set; we call these vertices *interface vertices*, and call the ones in the other vertex set *latent vertices*. We will call the corresponding functions indexed by these two vertex sets *interface functions* and *latent functions* respectively, although we will be quite loose in speaking of a vertex and a function exchangeably as we do for a variable and an edge/half edge. We will customarily denote the set of interface vertices by I and the set of latent vertices by J. [1]

Note that since each interface function has exactly one half edge incident on it, unless it is more convenient to make the distinction, we will subsequently identify the set of half edges using

[1]We note that demanding no half edge incident on the latent functions entails no loss of generality, since if there is such a half edge, one may always insert a bivariate equality indicator function (or equivalently a bivariate max indicator or sum indicator) into the half edge, which converts the NFG to an equivalent one with this half edge turned into a regular edge. Since the bivariate equality indicator is both a split function and a conditional function, inserting such a function has no impact on our later restriction on the interface functions, where we require them to be all split functions or all conditional functions.

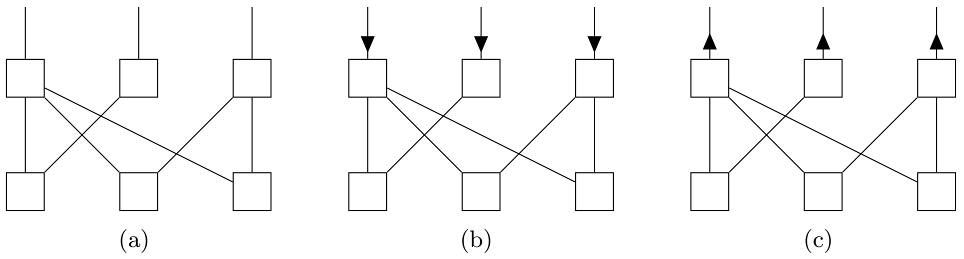


Figure 6: (a) an NFG model (b) a constrained NFG model (c) a generative NFG model

I, i.e., an interface vertex will index both its function and the half edge incident on it. An example NFG model is shown in Figure 6 (a), where the top-layer vertices are the interface vertices, and the bottom-layer vertices are the latent vertices. If necessary, we may formally denote such NFG using such a notation as $\mathcal{G}(I \cup J, E, f_{I \cup J})$.

In this modelling framework, we will focus on two "dual" families of models, which we call the *constrained NFG models* and the *generative NFG models* respectively (see, e.g., Figure 6 (b) and (c) respectively for a quick preview). We will demonstrate how these models are related to the previous models such as FG and CFG. We will also introduce "transformed NFG models" in Section 5, a special case of which reduces to CDN.

4.1 Constrained NFG model

A *constrained* NFG model is an NFG model in which all interface functions are split functions via their respective external variables.

To bring more intuition into this definition, we first take a slight digression and show in the following lemma that it is possible to "shape" a distribution by "random rejection".

Lemma 3. *Let X be a RV with a probability distribution p_X, where X assumes its values from a finite set $\mathcal{X}$, and let h be a normalized non-negative real function on $\mathcal{X}$ with a non-empty support, where the normalization is in the sense that $\max_{x \in \mathcal{X}} h(x) = 1$. Draw x from p_X and accept it with probability $h(x)$ and reject it with probability $1 - h(x)$. If x is accepted, output x; otherwise repeat the process until some other x' is drawn and accepted. Denote the output random variable by Y. Then the probability distribution p_Y of Y is, up to scale, $h(y)p_X(y)$, for all $y \in \mathcal{X}$.*

Proof. Let Z be a $\{0, 1\}$ RV representing the random rejection in the lemma, i.e., a sample $x \in \mathcal{X}$ is rejected if $Z = 0$, accepted if $Z = 1$, and the probability that $Z = 1$ is $h(x)$. Then the statement $Y = y$ is equivalent to $(X, Z) = (y, 1)$, and we have $p_Y(y) = p(X = y)p(Z = 1|X = y) = p_X(y)h(y)$. □

The idea of "distribution shaping" via "random rejection" is central to the semantics of constrained NFG models, which we demonstrate in the example next.

Example 1. *Let $\mathcal{G}$ be a constrained NFG as in Fig. 7 where $f_1(x_1, s_1, s_1')$ and $f_2(x_2, s_2, s_2')$ are positive functions that split via x_1 and x_2, respectively, and h_1, h_2 and h_3 are non-negative functions (with non-empty supports). From Lemma 1 we may express, up to a respective scaling factor, f_1 as*

$p_{X_1}(x_1)p_{S_1|X_1}(s_1|x_1)p_{S'_1|X_1}(s'_1|x_1)$ and f_2 as $p_{X_2}(x_2)p_{S_2|X_2}(s_2|x_2)p_{S'_2|X_2}(s'_2|x_2)$, for some distributions p_{X_1}, p_{X_2}, $p_{S_1|X_1}$, $p_{S'_1|X_1}$, $p_{S_2|X_2}$ and $p_{S'_2|X_2}$. The RVs represented by the NFG may be regarded as being generated by the following process.

1. *Draw (x_1, x_2) from distribution $p_{X_1}(x_1)p_{X_2}(x_2)$ where p_{X_1} and p_{X_2} are as specified by our choices above. Note that the two components of the drawn vector are independent.*

2. *Draw vector (s_1, s'_1) from the distribution $p_{S_1|X_1}(s_1|x_1)p_{S'_1|X_1}(s'_1|x_1)$ and draw (s_2, s'_2) from $p_{S_2|X_2}(s_2|x_2)p_{S'_2|X_2}(s'_2|x_2)$. It is clear that the joint distribution of $(x_1, x_2, s_1, s'_1, s_2, s'_2)$ is up to scale $f_1(x_1, s_1, s'_1)f_2(x_2, s_2, s'_2)$.*

3. *Let $H(s_1, s'_1, s_2, s'_2) := c \cdot h_1(s_1)h_2(s'_1, s_2)h_3(s'_2)$, where c is a normalizing constant such that the maximum value of $H(\cdot)$ is 1. Accept the drawn vector $(x_1, x_2, s_1, s'_1, s_2, s'_2)$ with probability $H(s_1, s'_1, s_2, s'_2)$ and reject it with probability $1 - H(s_1, s'_1, s_2, s'_2)$.*

4. *If the drawn $(x_1, x_2, s_1, s'_1, s_2, s'_2)$ is rejected, repeat the procedure from step 1, until the drawn $(x_1, x_2, s_1, s'_1, s_2, s'_2)$ is accepted. By Lemma 3, the accepted vector has a distribution equal, up to scale, to $f_1(x_1, s_1, s'_1)f_2(x_2, s_2, s'_2)H(s_1, s'_1, s_2, s'_2)$.*

5. *Output (x_1, x_2). Then clearly the output vector has distribution that is up to scale the exterior function of the NFG.*

The procedure introduced in the example above generalizes in an obvious way to arbitrary constrained NFG models. Instead of precisely, but repetitively, stating the procedure for the general setting, we make the following remarks. The interface functions completely specify how the external variables are drawn and how the internal variables are drawn conditioned on the drawn external configuration. The drawn internal configuration then undergoes a "random rejection" according to the product of all latent functions. The external configuration giving rise to an accepted internal configuration then necessarily follows the distribution prescribed by the exterior function of the NFG.

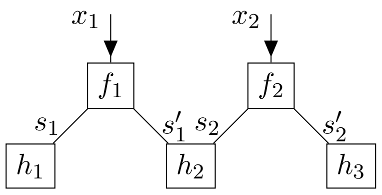


Figure 7: Example 1.

Analogously, one may view a constrained NFG model as a "probabilistic checking system": independent "inputs" (external variables) excite the "internal states" (internal variables) of the system via interface functions; the state configuration is "checked" probabilistically by the latent functions; only the external configurations that pass the internal check are kept. In general, the internal checking mechanism induces dependence among the external variables, which were *a priori* independent. In the special case when the latent functions are all indicator functions, the checking system is in fact deterministic, reducing to a set of constraints on the internal states, cf. Section 6.

This has been the motivation behind the name "constrained NFG model". As we will show momentarily that constrained NFG models and FG models are equivalent, the "probabilistic checking system" perspective of constrained models provides a different and new interpretation of the FG models.

4.2 Constrained NFG models are equivalent to FGs

Suppose that a constrained NFG model is such that every interface function is an equality indicator function. It is known [11] that one may convert such NFG to an FG according to the following procedure: *For each interface vertex, replace it by a variable vertex representing its half-edge variable and remove the half edge.*

Proposition 1. *If in a constrained NFG model all interface functions are equality indicators, then the above procedure gives rise to an FG equivalent to the NFG.*

Proof. Let $\mathcal{G}(I \cup J, E, f_{I \cup J})$ be the NFG in hand where $f_i = \delta_=$ for all $i \in I$. The resulting FG has an underlying graph $(I \cup J, E)$ where I and J are the variable and function indexing sets, respectively. Hence, the global function of the FG is the multiplication $\prod_{j \in J} f_j(x_{\mathrm{ne}(j)})$. On the other hand, if we use $T(v)$ to denote the set of internal edges incident on node v in the NFG, then the exterior function of the NFG is $\left\langle \prod_{j \in J} f_j(s_{T(j)}), \prod_{i \in I} \delta_=(x_i, s_{T(i)}) \right\rangle$, which, if i' and j' are connected by an edge t, accounts to substituting $x_{i'}$ in place of the argument s_t of $f_{j'}$ in the product $\prod_{j \in J} f_j(s_{T(j)})$, for all adjacent i' and j'. The claim follows by noting that $T(j) = \{\{i, j\} : i \in \mathrm{ne}(j)\}$. □

The proposition essentially suggests that the joint distribution represented by such a constrained NFG model factors multiplicatively and therefore can be represented by an FG. In fact the converse is also true, namely that any FG can be converted to an equivalent constrained NFG model with all interface functions being equality indicators. This is illustrated in Figure 8.

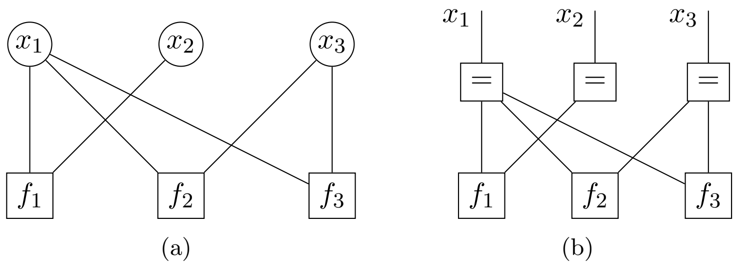


Figure 8: An FG and its equivalent constrained NFG.

Next we show that any constrained NFG is in fact equivalent to one with equality interface function. Given an arbitrary constrained NFG $\mathcal{G}$ where each interface function f_i splits as $\prod_{t \in T(i)} f_t$ for some bivariate functions f_t. The following procedure converts $\mathcal{G}$ into a constrained NFG with the same underlying graph as $\mathcal{G}$, and in which all interface functions are equality indicators:

1) Replace each interface function with an equality indicator.

2) Replace each hidden function f_j with $\langle f_j, f_t : t \in T(j) \rangle$.

Proposition 2. *In the above procedure, the original and resulting constrained NFGs are equivalent, and have the same underlying graph.*

Proof. The fact that the two NFGs have the same underlying graph is clear. To prove equivalence, each interface function f_i is the product $\prod_{t\in T(i)} f_t(x_i, s_t)$ of bivariate functions f_t, and from Proposition 1, it can be written as the sum-of-products form $\left\langle \prod_{t\in T(i)} f_t(s'_t, s_t), \delta_=(x_i, s'_{T(i)})\right\rangle$. Upon vertex splitting, each interface vertex can be replaced with the NFG representing its corresponding sum-of-product form. The claim follows upon vertex merging each hidden node f_j with its adjacent bivariate functions, i.e., by replacing f_j with $\left\langle f_j, f_t : t \in T(j)\right\rangle$. □

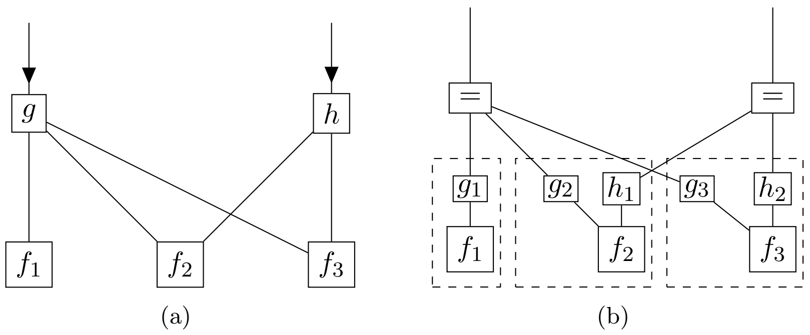


Figure 9: Converting a constrained NFG model to one in which all interface functions are equality indicators. (a) An example of a constrained NFG where by assumption g splits into g_1, g_2 and g_3, and h splits into h_1 and h_2, (b) vertex splitting of interface nodes, followed by vertex merging of each hidden function with its neighboring bivariate functions.

The conversion stated in the proposition and an illustration of the proof are shown in Figure 9. Invoking Proposition 1, the following theorem is immediate.

Theorem 1. *Every constrained NFG can be converted to an equivalent FG.*

4.3 Generative NFG model

A *generative* NFG model is an NFG model in which every interface function is a conditional function of its half edge variable given its remaining arguments. The following example gives sufficient insight of the modelling semantics of a generative NFG.

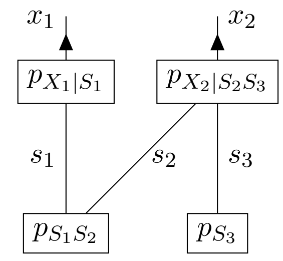


Figure 10: Example 2.

Example 2. *Let (S_1, S_2) be jointly distributed according to $p_{S_1 S_2}$ and S_3 be distributed according to p_{S_3}, where (S_1, S_2) is independent of S_3. Suppose that (X_1, X_2) depends on (S_1, S_2, S_3) according to conditional distribution $p_{X_1 X_2|S_1 S_2 S_3}(x_1, x_2|s_1, s_2, s_3) := p_{X_1|S_1}(x_1|s_1)p_{X_2|S_2 S_3}(x_2|s_2, s_3)$, it is then easy to verify that the joint distribution $p_{X_1 X_2}(x_1, x_2)$ is given by the sum-of-products form $\langle p_{S_1 S_2}, p_{S_3}, p_{X_1|S_1}, p_{X_2|S_2 S_3}\rangle$, and hence, the RVs (X_1, X_2) are represented by the generative NFG in Fig. 10.*

The NFG in this example is a generative NFG model where $p_{X_1|S_1}$ and $p_{X_2|S_2 S_3}$ are interface functions and $p_{S_1 S_2}$ and p_{S_3} are latent functions. In this case, the latent functions serve as independent sources of randomness, which "generate" the internal RVs (S_1, S_2 and S_3). The internal RVs then "generate" the external RVs via the interface functions.

In an arbitrary NFG model, since every latent function may be viewed as the joint distribution of its involved internal RVs, subject to a scaling factor, they can be regarded as independent "generating sources"; since each interface function is a conditional function, or, up to a scale, a conditional distribution of the external RV given its internal RVs, the product of these conditional functions may be regarded as the conditional distribution of all external RVs conditioned on the internal RVs. The product of all local functions is then up to scale the joint distribution of all external and internal RVs. The semantics of NFG then dictates that the joint distribution shall be summed over all internal variables, and the resulting exterior function is therefore the distribution of the external RVs, up to scale. In a sense, a generative NFG model describes how the external random variables are generated from some independent hidden sources.

4.4 A subclass of generative NFG models is equivalent to CFGs

In this section, we rely on the Fourier transform in some of the discussions. Let $\mathcal{X}$ be a finite abelian group, we use $\mathcal{X}^\wedge$ to denote the character group (written additively) of $\mathcal{X}$, defined as the set of homomorphisms from $\mathcal{X}$ to $\mathbb{C}$. It is well known that $(\mathcal{X}^\wedge)^\wedge$ is isomorphic to $\mathcal{X}$, and for any $x \in \mathcal{X}$ and $\hat{x} \in \mathcal{X}^\wedge$, $x(\hat{x}) = \hat{x}(x)$. We use $\kappa_{\mathcal{X}}(x, \hat{x})$ to denote both $x(\hat{x})$ and $\hat{x}(x)$, and use $\hat{\kappa}_{\mathcal{X}}(x, \hat{x})$ to denote $\kappa(x, -\hat{x})/|\mathcal{X}|$. For any function on $\mathcal{X}$, we define its Fourier transform as the sum-of-product form $\widehat{f}(\hat{x}) = \langle \kappa_{\mathcal{X}}(x, \hat{x}), f(x)\rangle$, for all $\hat{x} \in \mathcal{X}^\wedge$. It is not hard to show that κ and $\hat{\kappa}$ are an inverse-pair, and hence given $\widehat{f}$, one may recover f using the Fourier inverse as $f(x) = \langle \widehat{f}(\hat{x}), \hat{\kappa}(x, \hat{x})\rangle$, for all $x \in \mathcal{X}$. It is well known that if $\mathcal{X}$ is the direct product $\prod_{i \in I} \mathcal{X}_i$ of the finite abelian groups $\mathcal{X}_i$, then $(\mathcal{X})^\wedge$ is the direct product $\prod_{i \in I} \mathcal{X}_i^\wedge$, and it follows that $\kappa_{\mathcal{X}}(x, \hat{x}) = \prod_{i \in I} \kappa_{\mathcal{X}_i}(x_i, \hat{x}_i)$, for all $(x, \hat{x}) \in \mathcal{X} \times \mathcal{X}^\wedge$, and similarly for $\hat{\kappa}_{\mathcal{X}}$.

Suppose that a generative NFG model is such that every interface function is a sum indicator function. We may convert such an NFG to a CFG according to the following procedure: *For each interface vertex, replace it by a variable vertex representing its half-edge variable and remove the half edge.*

Proposition 3. *If in a generative NFG model all interface functions are sum indicators, then the above procedure gives rise to a CFG equivalent to the NFG.*

Proof. The proof follows the following steps: 1) Modify the NFG by replacing each interface function with a parity check indicator and inserting a degree two parity check indicator (a sign inverter) on each half edge, Fig. 11 (b). (This does not alter the exterior function due to the relation between the sum and the parity indicator function.) 2) Perform a holographic transformation on the resulting NFG by inserting the inverse-pair $\kappa_{\mathcal{X}_e}$ and $\hat{\kappa}_{\mathcal{X}_e}$ into each regular edge e (with $\kappa_{\mathcal{X}_e}$ adjacent to a hidden function or an inserted sign inverter), and inserting the transformers $\kappa_{\mathcal{X}_e}$ into each dangling edge e, Fig. (c). 3) By noting that (up to a scaling factor[2]) the Fourier and Fourier inverse of δ_+ are $\delta_=$, we obtain (after deleting all degree-two equality indicators resulting from the sign inverters) a constrained NFG in which each interface function is an equality indicator and each hidden function is the Fourier transform of the corresponding hidden function in the original NFG, Fig. (d). Hence, from the GHT and Proposition 1, we have $\widehat{Z}_{\mathcal{G}}(\hat{x}_I) = \prod_{j \in J} \widehat{f}_j(\hat{x}_{\mathrm{ne}(j)})$, and the claim follows from the multivariate multiplication-convolution duality theorem under the Fourier transform [6]. $\square$

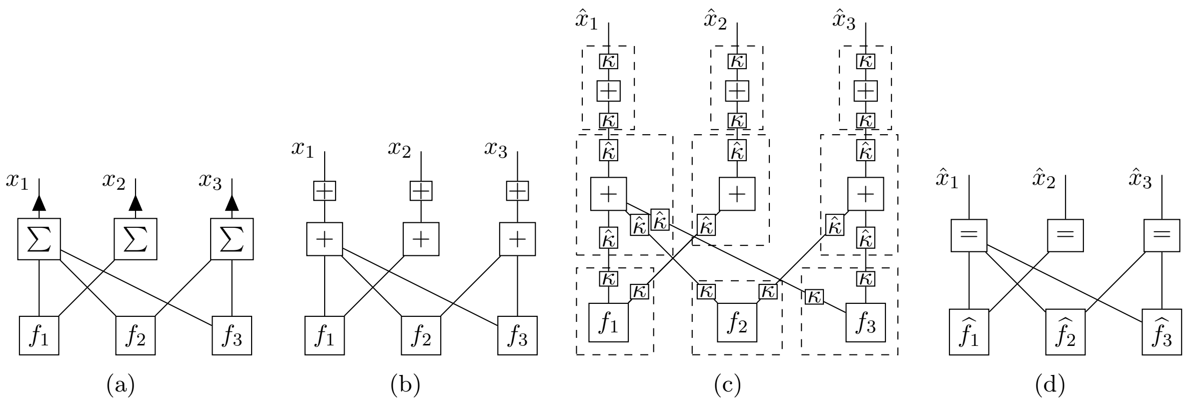


Figure 11: Proof of Proposition 3: (a) An example NFG, (b) Step 1, (c) Step 2, and (d) Step 3.

An example illustrating the steps of the proof is shown in Fig. 11. Of course one may attempt to prove the claim by direct evaluation of the exterior function as demonstrated in the following example.

Example 3. *Consider the NFG in Figure 10, where the interface functions are taken as sum indicators. Then the exterior function of this NFG is* $\sum_{s_1,s_2,s_3} p_{S1S2}(s_1,s_2)p_{S_3}(s_3)\delta_{\sum}(x_1,s_1)\delta_{\sum}(x_2,s_2,s_3) \overset{(a)}{=} \sum_{s_2,s_3} p_{S_1S_2}(x_1,s_2)p_{S3}(s_3)\delta_{\sum}(x_2,s_2,s_3) \overset{(b)}{=} p_{S_1S_2}(x_1,x_2) * p_{S3}(x_2)$*, where (a) identifies the bivariate sum indicator with equality indicator and removes it, and (b) is due to Lemma 2. The reader is invited to examine the structure of the original NFG and that of the CFG representing the above convolutional factorization.*

Indeed, for any generative NFG model in which interface functions are all sum indicators, the procedure above Proposition 3 applied to an interface function is equivalent to either applying step (a) above (for degree-2 vertices) or applying Lemma 2 (for vertices of degree higher than 2).

It is easy to see that the this procedure is reversible, in the sense that one may apply it in reverse direction and convert any CFG to a generative NFG model with all interface functions being sum indicators. Figure 12 shows an equivalent pair of CFG and generative NFG model.

4.5 Independence

We now show that there exists a "duality" between a constrained NFG model and a generative NFG model in their implied independence properties.

[2]It is not hard to show that all the scaling factors cancel out, and hence, all subsequent equalities are exact.

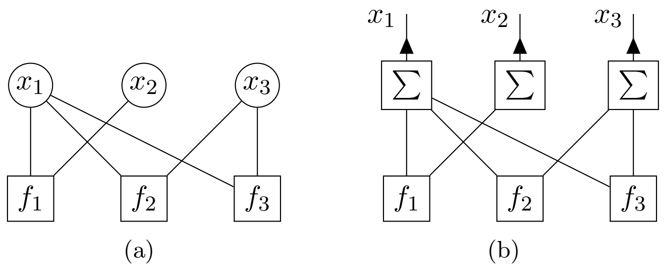


Figure 12: An equivalent pair of CFG (left) and generative NFG model (right).

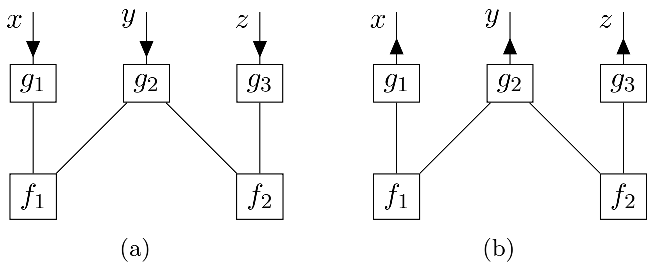


Figure 13: (a) $X \perp\!\!\!\perp Z|Y$ and (b) $X \perp\!\!\!\perp Z$.

Lemma 4. *For the NFG models in Fig. 13, we have $X \perp\!\!\!\perp Z|Y$ in the constrained model and $X \perp\!\!\!\perp Z$ in the generative model.*

Proof. For the constrained NFG, it is sufficient to show that $p(x, y, z)$ is a split function via y, see e.g. [15]. To this end, we have g_2 is a split function via y, say it splits into bivariate functions $g_{2,1}$ and $g_{2,2}$. Hence, applying the vertex splitting procedure for g_2 followed by the vertex merging of each hidden function and its adjacent bivariate function, it becomes clear that $p(x, y, z) = f_1'(x, y)f_2'(y, z)$ where $f_1' = \langle f_1, g_1, g_{2,1}\rangle$ and $f_2' = \langle f_2, g_3, g_{2,2}\rangle$.

For the generative model, we prove the claim graphically in Fig. 14. Marginalizing y, the probability distribution $p(x, z)$ is realized by the NFG in Fig. 14 (a), which, by the definition of a conditional function, is equivalent to the one in (b). Marginalizing again, we obtain the NFGs in (c) and (d) for the marginals $p(x)$ and $p(z)$, respectively. Hence, the multiplication $p(x)p(z)$ is realized by the NFG in (e), which is equivalent (again by the definition of a conditional function) to the one in (f). Noting that the NFG in the left side of (f) realizes the scalar 1, and comparing with (a), we see that $p(x, z)$ and $p(x)p(z)$ are realized by the same NFG, and hence, must be equal. □

We remark that the conditional independence part of the lemma can be proved graphically in a similar manner to the marginal independence part where marginalization is replaced with evaluation. Further, it is interesting to observe that the two NFG models have identical graphs, but the constrained model implies conditional independence property whereas the generative model implies the "dual" marginal independence property.

In an NFG model, suppose that $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{S}$ are three disjoint subsets of vertices. We say $\mathcal{A}$ and $\mathcal{B}$ are *separated* by $\mathcal{S}$ if every path from a vertex in $\mathcal{A}$ to a vertex in $\mathcal{B}$ go through some vertex in

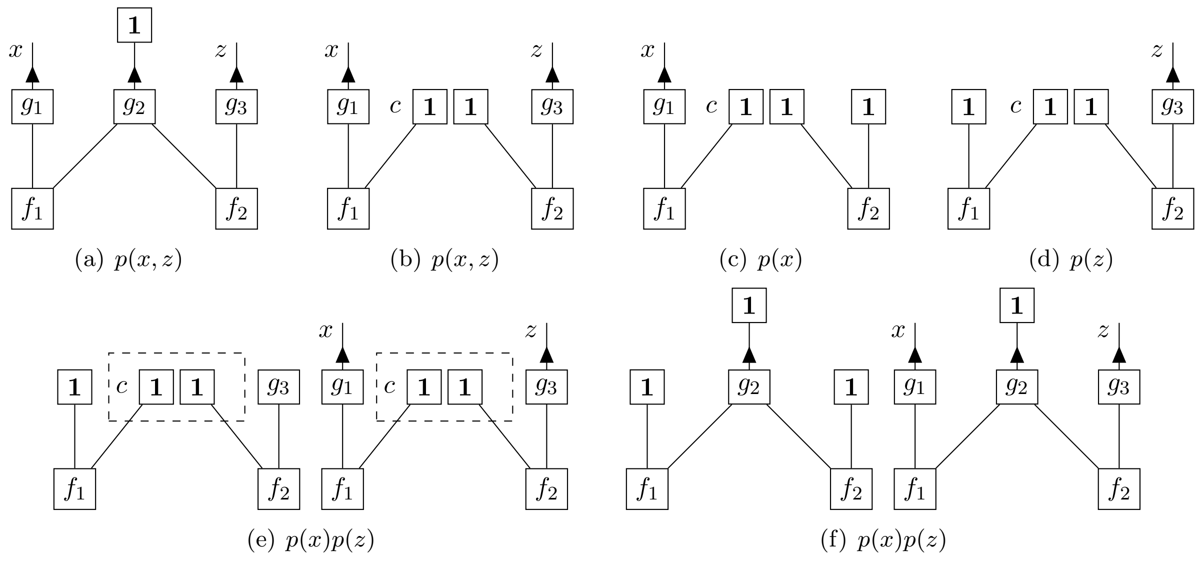


Figure 14: Proof of Lemma 4 for the generative model.

$\mathcal{S}$. In this case, if $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{S}$ are all subsets of the interface vertex set I, then, recalling that every external variable is also indexed by the interface vertex it is incident with, we also say that RV sets $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ are separated by the RV set $X_{\mathcal{S}}$. For any subset $I' \subseteq I$, let $\mathrm{ne}(I') := \{\mathrm{ne}(i) : i \in I'\}$. By merging the vertices in $\mathcal{A}$ into one vertex, and similarly for the interface nodes $\mathcal{S}$ and $\mathcal{B}' := I \backslash (\mathcal{A} \cup \mathcal{S})$, and performing the same merging for the hidden nodes $\mathrm{ne}(\mathcal{A})$ and $J \backslash \mathrm{ne}(\mathcal{A})$. Then the resulting NFG has the same graph topology as the ones in Fig. 13, as it is clear from the separation property that $\mathrm{ne}(\mathcal{B}') \subseteq J \backslash \mathrm{ne}(\mathcal{A})$. From the fact that the split and conditional properties are preserved under such mergings, the previous lemma extends in a straightforward manner to any NFG model, and we have the following theorem.

Theorem 2. *Let $\mathcal{G}(I \cup J, E, f_{I \cup J})$ be an NFG model and $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{S}$ be three disjoint interface vertex subsets, i.e., subsets of I. Suppose that $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ are separated by $X_{\mathcal{S}}$. Then*

1. *If the NFG is a constrained NFG model, then $X_{\mathcal{A}} \perp\!\!\!\perp X_{\mathcal{B}} | X_{\mathcal{S}}$.*
2. *If the NFG is a generative NFG model, then $X_{\mathcal{A}} \perp\!\!\!\perp X_{\mathcal{B}}$.*

Part 1 of Theorem 2 is essentially the global Markov property (see, e.g., [15]) on an FG model (noting that constrained NFG models are equivalent to FGs). Part 2 of the theorem, in the special case when all interface functions are sum indicators, was proved in [5] in the context of CFGs (noting that such NFG models reduce to CFGs). That is, Part 2 of Theorem 2 generalizes such a result from CFGs to arbitrary generative NFG models. We now provide some insights for this result.

Consider the NFG in Figure 13 (b). The fact $X \perp\!\!\!\perp Z$ can be reasoned by the fact that latent functions f_1 and f_2, giving rise to X and Z respectively, serve as independent sources of randomness. Indeed, it is precisely due to X and Z sharing no common latent functions that when Y is ignored X and Z become independent. The same is true for arbitrary generative NFG models, where if $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ are separated by $X_{\mathcal{S}}$, then we necessarily have $X_{\mathcal{A}}$ and $X_{\mathcal{B}}$ share no common latent functions.

We remark that the marginal independence, i.e., Part 2 of Theorem 2 holds for the more general class of NFG models characterized by the property that for each interface function f_i, it holds that

$$\sum_{x_i} f_i(x_i, x_{T(i)}) = \prod_{t \in T(i)} f_t(x_t),$$

for some univariate functions f_t. We may refer to an NFG model whose interface functions satisfy this property as an *extended generative model*, and it is clear that a generative model is an extended generative model. (A conditional function trivially satisfies the property above.) It is not hard to show that the class of constrained models and the class of extended generative models are closed under internal holographic transformations, from which, it follows that the independence properties in Theorem 2 are invariant under internal holographic transformations.

5 Transformed NFG Models

In some applications, instead of modelling the joint probability distribution of the RVs, we may wish our model to represent a certain transformation of the joint distribution, and it becomes clear that the NFG modelling framework introduced in this paper is particularly convenient for this purpose. Moreover, this framework provides a generic transformation technique and enables an infinite family of such transformations. In subsequent discussions, a *transformed NFG model*, or simply a transformed model, refers to any NFG obtained from an NFG model (generative or constrained) by a holographic transformation, where the external transformers in Step (H1) of the holographic transformation are not necessarily trivial— At some places we may refer to the original NFG as the *base model*.

Next we show that a particular class of NFG models, upon an appropriate choice of holographic transformation, results in CDNs. Let $\mathcal{X} := \{1, \ldots, |\mathcal{X}|\}$ and let bivariate function $A_{\mathcal{X}}$ on $\mathcal{X} \times \mathcal{X}$ be such that $A_{\mathcal{X}}(x, x') = 1$ if $x' \leq x$ and $A_{\mathcal{X}}(x, x') = 0$, otherwise. We call $A_{\mathcal{X}}$ a *cumulus* function. Let bivariate function $D_{\mathcal{X}}$ on $\mathcal{X} \times \mathcal{X}$ be such that $D_{\mathcal{X}}(x, x') = 1$ if $x = x'$, $D_{\mathcal{X}}(x, x') = -1$ if $x = x' + 1$, and $D_{\mathcal{X}}(x, x') = 0$ otherwise. We call $D_{\mathcal{X}}$ a *difference* function.

In the case where $\mathcal{X}$ is the Cartesian product $\prod_{i \in I} \mathcal{X}_i$ where $\mathcal{X}_i := \{1, \cdots, |\mathcal{X}_i|\}$ and I is a finite indexing set, then the previous definitions are extended to the partially-ordered set $\mathcal{X}$ in a component-wise manner by setting $A_{\mathcal{X}}(x_I, x'_I) := \prod_{i \in I} A_{\mathcal{X}_i}(x_i, x'_i)$ and $D_{\mathcal{X}}(x_I, x'_I) := \prod_{i \in I} D_{\mathcal{X}_i}(x_i, x'_i)$ for all $x_I, x'_I \in \mathcal{X}$. In our notations for cumulus and difference function vertices, we distinguish the first argument using a dot to mark the corresponding edge, cf. Fig. 15 (a).

The following lemma will be useful in characterizing CDNs as a subclass of transformed NFG models.

Lemma 5. *Let $\mathcal{X} = \prod_{i \in I} \mathcal{X}_i$ where $\mathcal{X}_i := \{1, \cdots, |\mathcal{X}_i|\}$ for all i in the finite indexing set I. Then,*
1. $\left\langle A_{\mathcal{X}}(x, y), D_{\mathcal{X}}(y, x') \right\rangle = \delta_{=}(x, x')$.
2. For any set of RVs X_I, $\left\langle p_{X_I}(x'), A_{\mathcal{X}}(x, x') \right\rangle$ is $F_{X_I}(x)$ for all $x \in \mathcal{X}$.
3. The two NFGs in Figure 15 are equivalent where each edge variable assumes its values from $\mathcal{X}$.

Proof. Parts 1 and 2 are immediate from the definitions of $A_{\mathcal{X}}$ and $D_{\mathcal{X}}$. For part 3, let $\mathcal{G}$ be as in Fig. 15 (a). First we prove the result for $n = 3$ and $|I| = 1$, i.e., $\mathcal{X} = \{1, \cdots, |\mathcal{X}|\}$, where by the

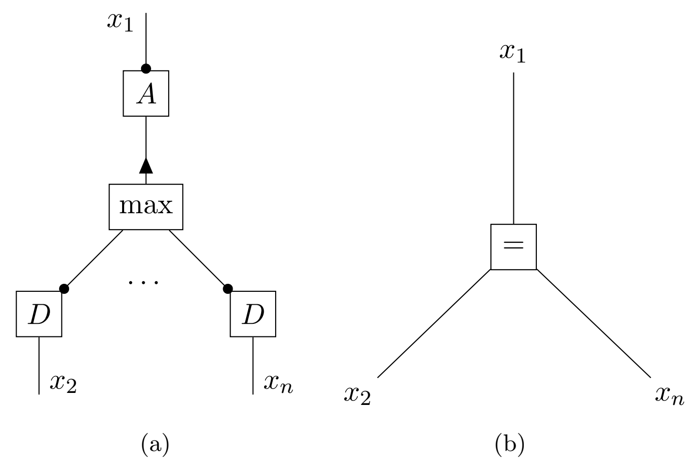


Figure 15: Two equivalent NFGs

definitions of the difference transform, the exterior function, and the max indicator function, we have

$$Z_{\mathcal{G}}(x_1,x_2,x_3)=\begin{cases} A_{\mathcal{X}}(x_1,\max(x_2,x_3))-A_{\mathcal{X}}(x_1,\max(x_2+1,x_3))- & \\ A_{\mathcal{X}}(x_1,\max(x_2,x_3+1))+A_{\mathcal{X}}(x_1,\max(x_2+1,x_3+1)) & ,x_2,x_3<|\mathcal{X}| \\ A_{\mathcal{X}}(x_1,\max(x_2,x_3))-A_{\mathcal{X}}(x_1,\max(x_2+1,x_3)) & ,x_2<|\mathcal{X}|,x_3=|\mathcal{X}| \\ A_{\mathcal{X}}(x_1,\max(x_2,x_3))-A_{\mathcal{X}}(x_1,\max(x_2,x_3+1)) & ,x_2=|\mathcal{X}|,x_3<|\mathcal{X}| \\ A_{\mathcal{X}}(x_1,\max(x_2,x_3)) & ,x_2,x_3=|\mathcal{X}| \end{cases}$$

From the definition of the cumulus, it is clear that any possible non-zero values of $Z_{\mathcal{G}}$ may occur only if $x_1 \geq \max(x_2,x_3)$. Assume $x_1 > \max(x_2,x_3)$ and note that in this case it is impossible to simultaneously have $x_2=|\mathcal{X}|$ and $x_3=|\mathcal{X}|$, then

$$\begin{aligned} Z_{\mathcal{G}}(x_1,x_2,x_3) &= \begin{cases} 1-1-1+1 & ,x_2,x_3<|\mathcal{X}| \\ 1-1 & ,x_2<|\mathcal{X}|,x_3=|\mathcal{X}| \\ 1-1 & ,x_2=|\mathcal{X}|,x_3<|\mathcal{X}| \end{cases} \\ &= 0. \end{aligned}$$

Hence, assume $x_1=\max(x_2,x_3)$ and further suppose $x_2<x_3$, then

$$\begin{aligned} Z_{\mathcal{G}}(x_1,x_2,x_3) &= \begin{cases} 1-1 & ,x_2,x_3<|\mathcal{X}| \\ 1-1 & ,x_2<|\mathcal{X}|,x_3=|\mathcal{X}| \end{cases} \\ &= 0. \end{aligned}$$

By symmetry, we also have $Z_{\mathcal{G}}(x_1,x_2,x_3)=0$ for $x_2>x_3$. The only possibility left is $x_2=x_3$, in which case, it is clear that $Z_{\mathcal{G}}(x_1,x_2,x_3)=1$. For $|I|>1$, the claim follows from the fact that the cumulus, difference, max and equality functions are defined in a component-wise manner.

For general n, using the fact that $\max(x_2,x_3,\dots,x_n)=\max(x_2,\max(x_3,\dots\max(x_n-1,x_n)\dots))$, we can express the NFG in Fig. 15 (a) using the NFG in Fig. 16 (a). Inserting the inverse-pair $A_{\mathcal{X}}$ and $D_{\mathcal{X}}$ on each edge inside the dashed box in Fig. 16 (a), we obtain the equivalent NFG in (b). Invoking the established part of the lemma for $n=3$ under the vertex merging shown in (b), and the claim follows. □

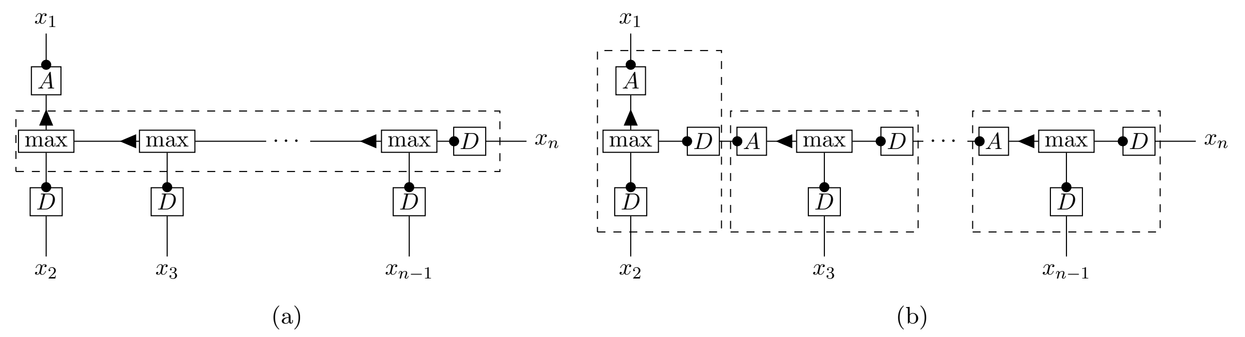


Figure 16: Proof of Part 3 of Lemma 5 for arbitrary n.

In this lemma, Part 1 suggests that $A_{\mathcal{X}}$ and $D_{\mathcal{X}}$ are an inverse-pair transformers, and Part 2 suggests that cumulus functions may serve to transform a probability distribution to a CDF.

Given a generative NFG in which each interface function is a max indicator and each hidden function is a probability distribution of the variables incident on it. Let $\mathcal{G}$ be the transformed model obtained from such generative NFG by inserting the cumulus transformer $A_{\mathcal{X}_i}$ on each half-edge. The following procedure converts $\mathcal{G}$ into a CDN:

1) Replace each max indicator and its adjacent transformer with a variable node representing the transformer's half-edge variable, and delete the half edge.

2) Replace each hidden function, which is a probability distribution, with the corresponding cumulative distribution.

Theorem 3. *If in a transformed model all external transformers are cumulus transformers, all interface functions are max indicators, and all hidden functions are probability distributions, then the above procedure gives rise to a CDN equivalent to the transformed model.*

Proof. Perform a holographic transformation on the transformed model by inserting into each internal edge e the inverse-pair transformers $A_{\mathcal{X}_e}$ and $D_{\mathcal{X}_e}$, with the cumulus facing the hidden function and the difference transformer facing the max indicator. Merging each hidden node with its neighboring cumulus transformers, by Part (2) of Lemma 5, the resulting node represents the desired CDF. Merging each max indicator with its neighboring difference transformers and the already existing cumulus, by Part (3) of Lemma 5, we arrive to a constrained NFG in which each interface function is an equality indicator, and the claim follows by Proposition 1. □

Figure 17 demonstrates the proof on an example NFG.

Before we proceed, we remark that the independence properties implied by a transformed model are precisely the ones implied by its base NFG, i.e., by ignoring all the external transformers. By the remark succeeding Theorem 2, such independence properties are further invariant under internal holographic transformations. Since the base NFG of the transformed model in Theorem 3 is a generative model, it becomes clear that the independence properties implied by CDNs are the marginal independence ones, i.e., Part 2 of Theorem 2.

Besides CDNs, we also remark that it is possible to regard linear characteristic models (LCMs) [7] as a special case of transformed NFG models. In fact, the NFG in Fig. 11 (d) is a transformed model that is equivalent to an LCM, and the holographic transformation presented in the proof of

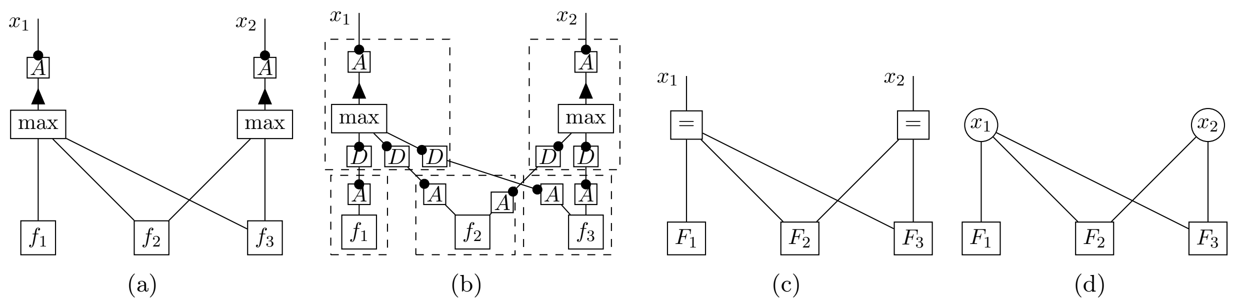


Figure 17: (a) An example of a transformed model in accordance to Theorem 3. (b) Inserting the inverse-pair cumulus-difference transformers on each internal edge, and vertex merging each node with its neighboring transformers, by the GHT, the resulting NFG is equivalent to the one in (a). (c) By Lemma 5 an equivalent NFG to the one in (b), where F_i is the cumulus transform of f_i, which is a CDF if f_i is a probability distribution. (d) By Proposition 1, an equivalent CDN to the NFG in (c).

Proposition 3, Fig. 11 (c), provides the basis for understanding an LCM as a transformed model whose base NFG is a generative NFG equivalent to a CFG, Fig. 11 (a). We skip details, and hope the framework of transformed NFG models is clear enough to see such equivalence.

6 Linear codes

In this section we discuss the connection between NFG models and linear codes. The material in this section is well-known [16], yet, we choose to address it in this work in light of the constrained and generative semantics.

Any code $\mathcal{C}$, linear or not linear, can be described in terms of its *membership* function, namely, the indicator function $\delta_{\mathcal{C}}(y) := [y \in \mathcal{C}]$ for all $y \in \mathcal{X}$ and for some finite set $\mathcal{X}$. Clearly $\delta_{\mathcal{C}}$ may be viewed, up to scaling factor, as a distribution function over $\mathcal{X}$ and in the case of linear codes, as we will see, may be described in terms of a constrained or a generative NFG model.

A *linear code* of length n and dimension k over a finite field $\mathbb{F}$ is a k dimensional subvector space of $\mathbb{F}^n$. Classically, a linear code $\mathcal{C}$ can be expressed in two dual ways. At one hand, $\mathcal{C} = \{f(x) : x \in \mathbb{F}^k\}$ for some linear function $f : \mathbb{F}^k \to \mathbb{F}^n$. This can also be written as $\mathcal{C} = \{\big(f_1(x), \ldots, f_n(x)\big) : x \in \mathbb{F}^k\}$ for some linear maps $f_i : \mathbb{F}^k \to \mathbb{F}$ for all i. That is, the code indicator function $\delta_{\mathcal{C}}(y)$ for all $y \in \mathbb{F}^n$, can be expressed as the sum-of-products form

$$\delta_{\mathcal{C}}(y_1, \ldots, y_n) = \sum_{x \in \mathbb{F}^k} \prod_{i=1}^{n} [y_i = f_i(x)],$$

for all $y_i \in \mathbb{F}$. Clearly, each local function $[y_i = f_i(x)]$ is a conditional function of y_i given x, and so, the indicator function of $\mathcal{C}$ can be realized using a generative NFG model. In fact, since each linear function $f_i : \mathbb{F}^k \to \mathbb{F}$ can be written as $f_i(x_1, \ldots, x_k) = a_{i1}x_1 + \ldots + a_{ik}x_k$ for some $a_{ij} \in \mathbb{F}$, it follows that each interface function is a sum indicator function involving y_i and x_j for all j such that $a_{ij} \neq 0$. The role of the hidden functions is to provide replicas of each variable x_j for all $j \in \{1, \cdots, k\}$, and hence are taken as equality indicators. More explicitly, for each x_j, we have a hidden equality indicator of degree equals the number of interface functions involving x_j. This guarantees that each variable appears in the desired number of interface functions while respecting the degree restrictions [11]. From this we arrive to a generative NFG with n interface nodes (each is a sum indicator), k hidden nodes (each is an equality indicator), and there is an edge connecting nodes i and j if and only if a_{ij} is nonzero. Fig. 18 (a) illustrates the generative NFG model for the Hamming code (with $n = 7$ and $k = 4$).

On the other hand, the parity check interpretation of a linear code dictates that the elements of a linear code $\mathcal{C}$ must satisfy a collection of homogeneous linear equations. That is, $\mathcal{C} = \{y \in \mathbb{F}^n : f(y) = 0\}$ for some linear map $f : \mathbb{F}^n \to \mathbb{F}^{n-k}$. This can also be written as $\mathcal{C} = \{y \in \mathbb{F}^n : (f_1(y), \ldots, f_{n-k}(y)) = 0\}$ for some linear maps $f_j : \mathbb{F}^n \to \mathbb{F}$. Hence, the code indicator function can be expressed as the product

$$\delta_{\mathcal{C}}(y) = \prod_j [f_j(y) = 0],$$

which (since f_j is linear) can further be simplified as $\delta_{\mathcal{C}}(y_1, \ldots, y_n) = \prod_j [\sum_i a_{ij} y_i = 0]$ for some $a_{ij} \in \mathbb{F}$. From Proposition 1, we can see that the code indicator function is realized by a constrained NFG model in which each interface node i is an equality indicator, each hidden node j is a parity indicator, and there is an edge connecting nodes i and j if and only if a_{ij} is nonzero, Fig. 19 (a).

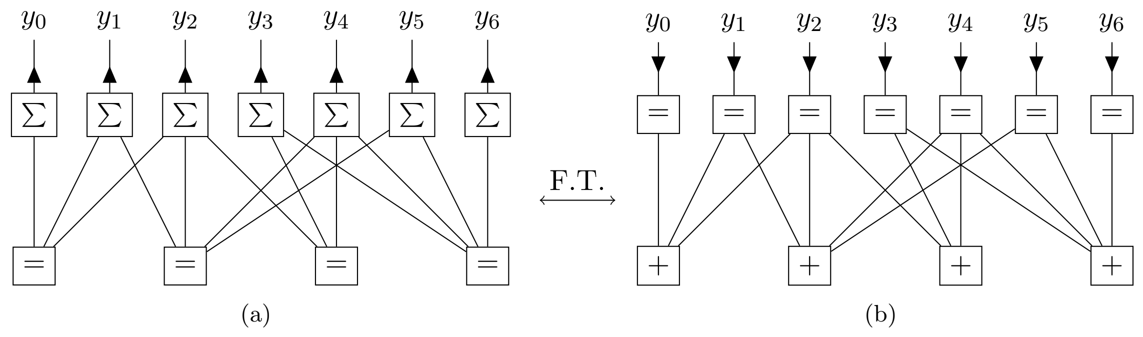


Figure 18: (a) Generator realization of Hamming code, and (b) parity realization of dual Hamming code.

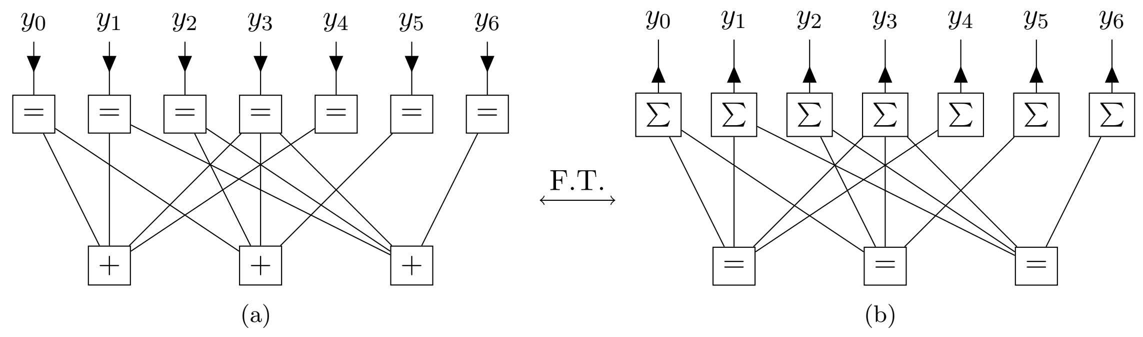


Figure 19: (a) Parity realization of Hamming code, and (b) generator realization of dual Hamming code.

In summary, a constrained NFG model of a linear code represents a parity realization, and a generative NFG model represents a generator realization. From the duality between the sum indicator and the equality indicator under the Fourier transform, the duality between a generator realization of a code and a parity realization of the dual code may be explained as follows: Starting with a generative NFG model (generator realization) of a linear code, Fig. 18 (a), and performing a holographic transformation with Fourier transformers, one obtains a constrained NFG model (a parity realization) of the dual code, Fig. 18 (b). Conversely, starting with a constrained NFG model (a parity realization) of the code, Fig. 19 (a), one ends with a generative NFG model (a generator realization) of the dual code, Fig. 19 (b).

7 Evaluation of the exterior function

In this section we discuss the algorithmic aspect of evaluating the exterior function of NFGs. We start with the "elimination algorithm" for NFGs, which is essentially the well-known elimination algorithm of inference on undirected graphical models [17]. The elimination algorithm is exact but its complexity depends on the ordering at which the elimination is performed.[3] A more efficient algorithm, but exact only on NFGs with no cycles, is the sum-product algorithm [3], which we discuss in the language of NFGs [18] in the second part of this section. Finally, we discuss an "indirect approach" for evaluating the exterior function, where a holographic transformation, if it exists, is used to convert the NFG into one that is more appropriate for the such computation.

7.1 Elimination Algorithm

The following algorithm computes the exterior function of an arbitrary NFG (i.e. not necessarily a bipartite).

Algorithm 1 (Elimination)**.** *Given an NFG $\mathcal{G}$.*

```
While $\mathcal{G}$ is not a single node.  Do
{
Pick an adjacent pair of vertices $v_1$ and $v_2$ in $\mathcal{G}$;
Compute $f_{v_1v_2}(x_{E(v_1)\cup E(v_2)\setminus E(v_1)\cap E(v_2)}) := \langle f_{v_1}, f_{v_2}\rangle$;
Update $\mathcal{G}$ by removing $f_{v_1}$ and $f_{v_2}$, and adding $f_{v_1v_2}$;
}
```

Evidently, this algorithm runs in a finite time and terminates with a single node whose function is the desired exterior function. This is essentially the vertex merging procedure applied recursively on pairs of adjacent vertices. Clearly, the elimination algorithm may equivalently be viewed as an elimination algorithm on the edges of the NFG, where in each step it eliminates all the edges between a pair of adjacent vertices. More precisely, one may say, the elimination algorithm is a merging algorithm on the nodes, and is an elimination algorithm on the edges of the NFG. In subsequent discussions, we will freely alternate between such two views. Note that even if we start with a simple NFG, parallel edges may still arise[4] while applying the elimination algorithm. However, loops may never arise since in each step we eliminate all parallel edges at once.

Example 4. *Let $\mathcal{G}$ be as in Fig 20 (a). Applying the elimination algorithm, we obtain:*

[3]The problem of finding the "best" elimination ordering is known to be NP-hard, where the term "best" is in the sense of minimizing the largest node-degree (of nodes that do not factor multiplicatively) arising while performing the elimination algorithm, and the minimization is over all possible orderings.

[4]It is not hard to see that parallel edges do not appear at any step of the elimination algorithm if and only if the NFG is cycle-free, i.e., is a tree.

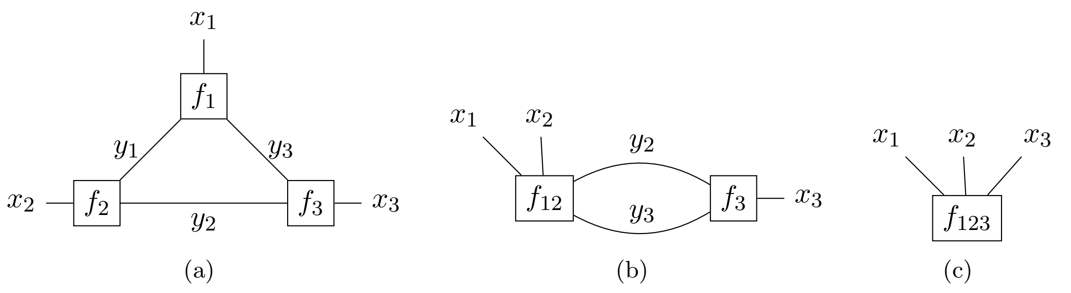


Figure 20: Elimination algorithm example.

Eliminating y_1 gives the NFG in Fig. (b) with $f_{12}(x_1, x_2, y_2, y_3) = \big\langle f_1(x_1, y_1, y_3), f_2(x_2, y_1, y_2)\big\rangle$, eliminating y_2, y_3 gives the NFG in Fig. (c) with $f_{123}(x_1, x_2, x_3) = \big\langle f_{12}(x_1, x_2, y_2, y_3), f_3(x_3, y_2, y_3)\big\rangle$, and it is easy to verify that $Z_{\mathcal{G}} = f_{123}$.

For any node v in an NFG, let $\deg(v) := |E(v)|$ be the number of external and internal edges incident on v. The following lemma determines the complexity of eliminating a pair of adjacent vertices in an NFG.

Lemma 6. *The complexity of eliminating a pair of adjacent vertices u, v in the elimination algorithm is of order $|\mathcal{X}|^{\deg(u)+\deg(v)-|E(u)\cap E(v)|}$.*

Proof. For each $x \in \mathcal{X}_{E(u)\cup E(v)\setminus E(u)\cap E(v)}$, we need $|\mathcal{X}_{E(u)\cap E(v)}|$ computations, and the claim follows by noting that $|E(u) \cup E(v)\setminus E(u) \cap E(v)| = |E(u)| + |E(v)| - 2|E(u) \cap E(v)|$. □

The following example shows that for some indicator functions of interest, the elimination complexity can significantly be reduced.

Example 5. *Consider the NFG in Fig. 21 (a), where each edge is associated the same alphabet $\mathcal{X}$. In general, the complexity of computing the exterior function is of order $|\mathcal{X}|^{n+1}$. In the special case where f is the indicator function $\delta_{\sum}$ or $\delta_{\max}$, then the complexity is $(n-1)|\mathcal{X}|^2$. This is because such indicator functions may further be factorized as shown in Figs. 21 (b) and (c). To see this, the elimination algorithm may start by eliminating $t_1, \ldots, t_n$, which induces no complexity as each elimination simply accounts to pointing to the proper entry of each function f_i, resulting in Fig. (d), where f_i' is properly defined according to $\delta_{\sum}$ or $\delta_{\max}$. Now eliminating each e_i (starting with e_{n-1}) costs $|\mathcal{X}|^2$ computations, giving rise to $(n-1)|\mathcal{X}|^2$ computations in total. Note that if $f = \delta_=$, then the complexity of computing the exterior function is $(n-1)|\mathcal{X}|$. (For each $x \in \mathcal{X}$, we need $(n-1)$ multiplications.)*

It is not hard to see that one may impose an elimination ordering such that the elimination algorithm may be viewed as one that merges a node with its neighbors, picks another node and merges it with its neighbors, and so on, until there is only one node left. We refer to such elimination as "block elimination," and to the elimination algorithm at the block level as the "block elimination algorithm," i.e., in each step, the block elimination algorithm replaces a function f_i and its neighbors with the function $\big\langle f_i, f_j : j \in \text{ne}(i)\big\rangle$. In the special case where each block eliminated function f_i is the equality indicator and none of the neighbors of f_i share any edges, cf. Fig. 22 and Example 6, then the block elimination becomes the classical elimination algorithm of undirected graphical models. (An undirected graphical model can be converted to a FG which by Proposition 1 is

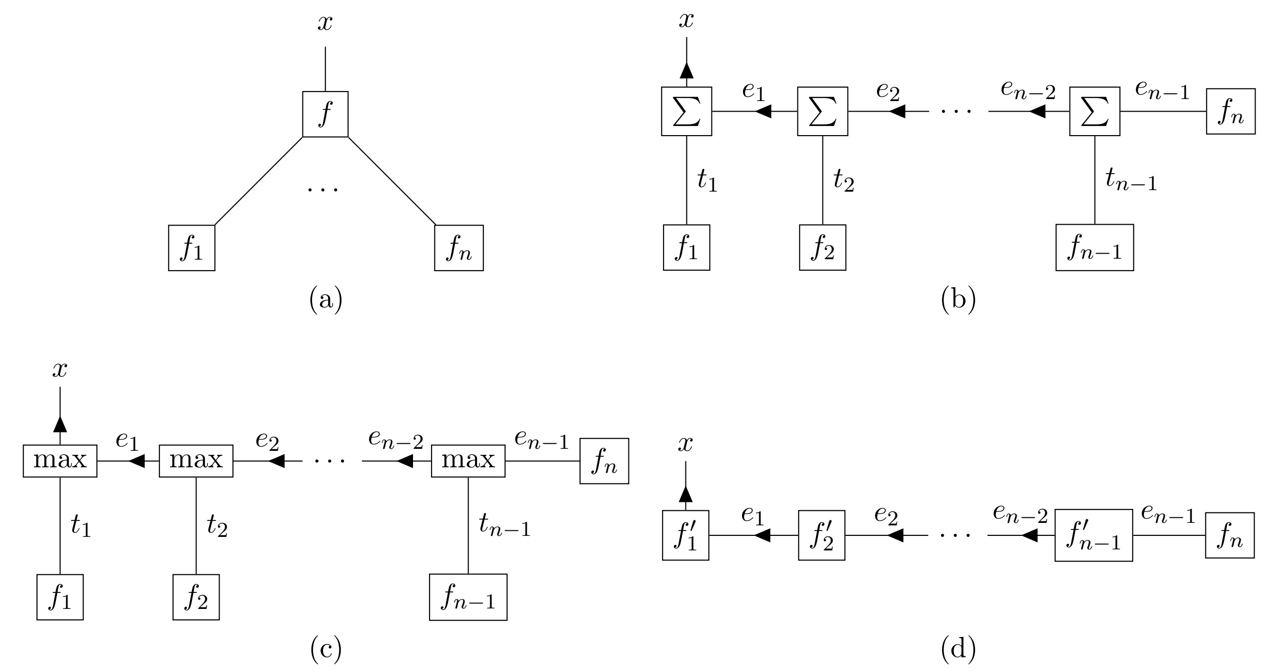


Figure 21: Example 5: The complexity of eliminating a sum or max indicator function.

equivalent to a constrained NFG whose interface functions are equality indicators— Note that none of the neighbors of such equality nodes shares any edges due to the bipartite nature of an NFG model.) We remark that whereas the elimination algorithm does not preserve the bipartite structure of an NFG in each edge elimination step, on the block elimination level, the algorithm respects such bipartite structure. (Let I and J be the two independent vertex sets, and let $f_{\mathrm{ne}(i)}$ be the node resulting from merging node $i \in I$ and its neighbors. Let node $j \in J$ be connected to $f_{\mathrm{ne}(i)}$ by some edge e, then $e \in E(i)$ or $e \in E(j')$ for some $j' \in \mathrm{ne}(i)$. Both such cases are impossible, since $e \in E(i)$ implies $j \in \mathrm{ne}(i)$ and so j would have been merged with i in the block elimination, and $e \in E(j')$ violates the bipartite assumption.)

The following example addresses the complexity of a block elimination.

Example 6. *Let $\mathcal{G}$ be as in Fig. 23, where $E_1, \ldots, E_n$ are disjoint. This NFG may be understood as a sub-NFG, the computation of its exterior function corresponds to a block elimination of a node and its neighbors in a bigger NFG. Note that if the bigger NFG is a bipartite, then the requirement that $E_1, \ldots, E_n$ be disjoint is automatically satisfied. Assuming the variable of each edge incident on f takes its values from $\mathcal{X}$, then by recursive application of Lemma 6, the complexity of computing the exterior function $Z_{\mathcal{G}}(x_{E_1}, \ldots, x_{E_n})$ is given by (assuming the elimination order $x_1, x_2, \ldots, x_n$)*

$$\sum_{i=1}^{n} |\mathcal{X}|^{1+n-i+\sum_{j=1}^{i} |E_j|}.$$

That is, if E_i is non-empty for all i, then the complexity is of order $|\mathcal{X}|^{1+|E_1|+\cdots+|E_n|}$. As an

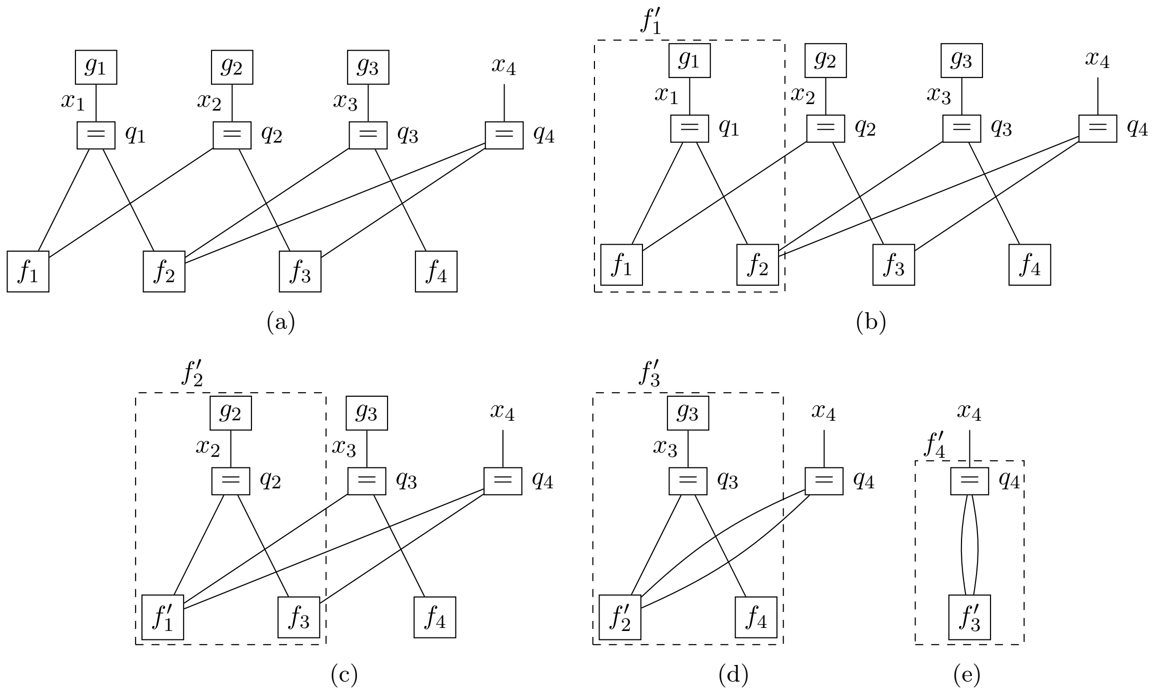


Figure 22: Block elimination algorithm: (a) an example NFG $\mathcal{G}$, (b), (c), (d), and (e) illustrate the block elimination of nodes q_1, q_2, q_3, and q_4, respectively, and it is not hard to see that $Z_{\mathcal{G}}(x_4) = f'_4(x_4) = f'_3(x_4, x_4)$. Note that if $\mathcal{G}$ is obtained from a constrained NFG by gluing functions g_1, g_2, and g_3 into the external edges x_1, x_2, and x_3 of the constrained NFG, then in the special case where every g_i is the constant-one indicator, it follows that $Z_{\mathcal{G}}$ is the marginal probability distribution p_{X_4}, cf. Section 8.

example, consider for instance $n = 3$, then the exterior function of $\mathcal{G}$ is given by

$$Z_{\mathcal{G}}(x_{E_1}, x_{E_2}, x_{E_3}) = \overbrace{\Big\langle f_3, \underbrace{\big\langle f_2, \underbrace{\langle f_1, f\rangle}_{\text{SP1}} \big\rangle}_{\text{SP2}} \Big\rangle}^{\text{SP3}}.$$

Computing the first sum of products (SP1), i.e., eliminating x_1, involves $|\mathcal{X}|^{|E_1|+3}$ computations by Lemma 6, and similarly, one needs $|\mathcal{X}|^{|E_1|+|E_2|+2}$ operations to compute $SP2$. Finally, SP3 requires $|\mathcal{X}|^{|E_1|+|E_2|+|E_3|+1}$ operations.

Next we consider the transformation of a function. Suppose we are interested in the exterior function of the NFG in Fig. 24 (a), where x and x' take their values from a finite alphabet $\mathcal{X}^n$ for some positive integer n. Clearly the exterior function represents a matrix-vector multiplication, and hence may be computed in $|\mathcal{X}|^{2n}$ operations. In the case where the bivariate function (on $\mathcal{X}^n \times \mathcal{X}^n$) f' factors as the product of bivariate functions (on $\mathcal{X} \times \mathcal{X}$) $f'_1, \cdots, f'_n$, then from the previous example, it follows that the complexity of transforming the function f via transformers $f'_1, \cdots, f'_n$ is $n|\mathcal{X}|^{n+1}$. (This is the special case with $|E_1| = \cdots = |E_n| = 1$, compare Figs 23 and

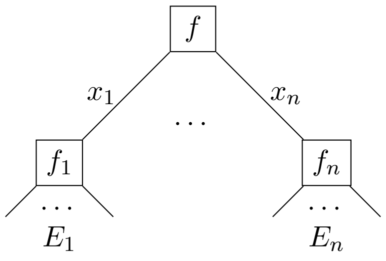


Figure 23: Example 6.

24 (b).) Another way of viewing this is that the elimination of each x_i accounts to a matrix-vector multiplication for a given configuration $x_{N\setminus\{i\}}$, where $N := \{1, \cdots, n\}$, and hence requires $|\mathcal{X}|^2$ computations. That is, eliminating x_i requires $|\mathcal{X}|^{(n-1)+2}$ operations, and eliminating $x_1, \cdots, x_n$ requires in total $n|\mathcal{X}|^{n+1}$ operations, as observed.

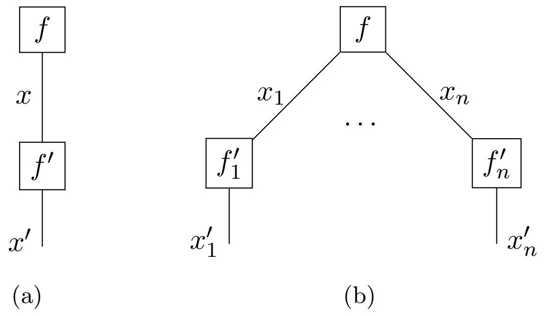


Figure 24: Transformation of a function.

We emphasize that if the transformers exhibit a special form, then more efficient computations may be achieved. For instance, if each f'_i is a Fourier kernel, then the fast Fourier transform may be used in the elimination of each edge using $\log(|\mathcal{X}|)|\mathcal{X}|^n$ computations, giving rise to a total $n\log(|\mathcal{X}|)|\mathcal{X}|^n$ operations for computing the transformation of f. Another important case where such savings are possible is when each f'_i is the cumulus or the difference transform. In such case, the matrix-vector multiplication associated with the elimination of each edge may be performed using $|\mathcal{X}|$ operations, giving rise to a total complexity of $n|\mathcal{X}|^n$. To see this, consider our example with $n = 3$ and assume f'_1, f'_2, f'_3 are cumulus transforms. To eliminate x_1, we start with the initiation $s := f$, and for $x'_1 = 2, \ldots, |\mathcal{X}|$, we update s as,

$$s(x'_1, x_2, x_3) = f(x'_1, x_2, x_3) + s(x'_1 - 1, x_2, x_3).$$

Hence, computing SP1 in $|\mathcal{X}|^3$ operations instead of $|\mathcal{X}|^4$. Similarly, we compute SP2 and SP3, giving rise to a total number of operations $3|\mathcal{X}|^3$. This clearly extends to any n, and below we provide two algorithms for fast computation of the cumulus and difference transformations, where to facilitate notation, we use J^- to denote the set $\{1, \ldots, j-1\}$ and J^+ to denote $\{j+1, \ldots, n\}$ for any $j \in \{1, \ldots, n\}$.

Algorithm 2 (Fast cumulus transform). `Initialize:` $s_0 = f$;

```
For $j = 1, \ldots, n\{$
```

```
  For each $(x'_{J^-}, x_{J^+}) \in \mathcal{X}^{n-1}$ {
    Initialize: $s_j(x'_{J^-}, x'_j = 1, x_{J^+}) = s_{j-1}(x'_{J^-}, x_j = 1, x_{J^+})$;
    For $x'_j = 2, \ldots, |\mathcal{X}|$ {
    $s_j(x'_{J^-}, x'_j, x_{J^+}) = s_{j-1}(x'_{J^-}, x'_j, x_{J^+}) + s_j(x'_{J^-}, x'_j - 1, x_{J^+})$;
    }
  }
}
Return $s_n$;
```

The difference transformation is performed in exactly the same manner, except for the updating rule:

Algorithm 3 (Fast difference transform). `Initialize:` $s_0 = f$;

```
For $j = 1, \ldots, n$ {
  For each $(x'_{J^-}, x_{J^+}) \in \mathcal{X}^{n-1}$ {
    Initialize: $s_j(x'_{J^-}, x'_j = 1, x_{J^+}) = s_{j-1}(x'_{J^-}, x_j = 1, x_{J^+})$;
    For $x'_j = 2, \ldots, |\mathcal{X}|$ {
    $s_j(x'_{J^-}, x'_j, x_{J^+}) = s_{j-1}(x'_{J^-}, x'_j, x_{J^+}) - s_{j-1}(x'_{J^-}, x'_j - 1, x_{J^+})$;
    }
  }
}
Return $s_n$;
```

7.2 Sum-product algorithm

Given an NFG $\mathcal{G}$ with no external edges, in many circumstances, for each edge e in the NFG, one may be interested in computing the *marginal exterior function* [18] defined as

$$Z_{\mathcal{G}}(x_e) := \sum_{x_{E \setminus \{e\}}} \prod_{v \in V} f_v(x_{E(v)}).$$

It is possible to compute such marginals using the elimination algorithm by imposing an elimination ordering such that x_e appears last. (More precisely, the elimination algorithm is slightly modified here such that when it reaches x_e it multiplies the functions of the two nodes incident with e without summing out x_e.) Although the elimination algorithm is exact for any NFG, it may be expensive to perform, as it is likely to produce nodes with large degrees. Further, in addressing the marginals problem above, the elimination algorithm is repeated for each marginal, giving rise to some redundant computations since most of the computations used for evaluating one of the marginals can be used in determining some other marginals. The *sum-product algorithm* (SPA) is an efficient alternative in the case of NFGs with no cycles. We refer the reader to [3] for an excellent exposure to the SPA on FGs, and to [11, 18] for its formulation on NFGs.

Let $\mathcal{G}$ be a tree NFG (i.e. an NFG whose underlying graph is a tree) with a vertex set V and an edge set E comprised entirely of internal edges, i.e. $\mathcal{G}$ has no external edges. To facilitate notation, if nodes u and v are neighbors, we use e_{uv} to denote the edge $\{u, v\}$— Note that $e_{uv} = e_{vu}$. The SPA defines a set of *messages* that are passed between adjacent nodes, where we use $\mu_{u \to v}$ to denote the message passed from node u to node v. The messages are governed by the following *update rule*

(Fig. 25):

$$\mu_{u\to v}(x_{e_{uv}}) = \sum_{x\in\mathcal{X}_{E(u)\setminus\{e_{uv}\}}} f_u(x, x_{e_{uv}}) \prod_{v'\in \mathrm{ne}(u)\setminus\{v\}} \mu_{v'\to u}(x_{e_{v'u}}),$$

where a node u sends its message to adjacent node v only if it has received the messages of all its other neighbors, and the algorithm terminates when every node has sent a message to all its neighbors. That is, the message $\mu_{u\to v}$ is the sum-of-products $\langle f_u, \mu_{v'\to u} : v' \in \mathrm{ne}(u)\setminus\{v\}\rangle$, and the complexity of such computation is $(\deg(u)-1)|\mathcal{X}|^{\deg(u)-1}$. In the special case when f_u is the indicator function $\delta_{\sum}$ or $\delta_{\max}$, it is not hard to see that the complexity becomes $(\deg(u)-2)|\mathcal{X}|^2$. Further, it is clear that when f_u is the equality indicator, the message sent to node v is simply the multiplication of the incoming messages from all other neighbors of u, i.e.,

$$\mu_{u\to v}(x_{e_{uv}}) = \prod_{v'\in \mathrm{ne}(u)\setminus\{v\}} \mu_{v'\to u}(x_{e_{uv}}),$$

and the complexity of computing such message is $(\deg(u)-2)|\mathcal{X}|$.

Note that upon termination, the SPA would have computed $2|E|$ messages with two messages per edge, and it is possible to show, due to the tree structure, that each marginal $Z_{\mathcal{G}}(x_e)$ is given as the product of the two messages carried by the edge e. That is, for any $e_{uv}\in E$, we have

$$Z_{\mathcal{G}}(x_{e_{uv}}) = \mu_{u\to v}(x_{e_{uv}})\mu_{v\to u}(x_{e_{uv}}).$$

We remark that although the SPA was formulated on NFGs with no external edges, this does not present a serious limitation, as in many applications one converts the external edges, if present, into regular ones by gluing the constant-one or the evaluation indicators. (For each external edge the choice of the indicator function depends on interest and whether such edge is observed as "evidence" or not, cf. Section 8.) However, if one insists on NFGs with external edges, then the SPA algorithm still works for such NFGs, and it is possible to show that in this case if L is the set of external edges, then for any internal edge $e_{uv}\in E$, the product $\mu_{u\to v}\mu_{v\to u}$ is equal to $Z_{\mathcal{G}}(x_{e_{uv}}, x_L)$ defined as $Z_{\mathcal{G}}(x_{e_{uv}}, x_L) := \sum_{x_{E\setminus(L\cup\{e_{uv}\})}} \prod_{v\in V} f_v(x_{E(v)})$, and hence, the exterior function is given as $Z_{\mathcal{G}}(x_L) = \sum_{x_{e_{uv}}} Z_{\mathcal{G}}(x_{e_{uv}}, x_L)$. However, in this case, the SPA might be expensive to perform since the size of each message increases every time it is passed by a node with a dangling edge, as the message accumulates the variable of such edge as an argument.

Below, we give a simple example illustrating the SPA.

Example 7 (SPA example)**.** *Given the NFG in Fig. 26, we have*

$$\begin{aligned}
\mu_{1\to 2}(y_1) &= f_1(y_1), & \mu_{4\to 3}(y_3) &= f_4(y_3),\\
\mu_{2\to 3}(y_2) &= \sum_{y_1} f_2(y_1,y_2)\mu_{1\to 2}(y_1), & \mu_{3\to 5}(y_4) &= \sum_{y_2,y_3} f_3(y_2,y_3,y_4)\mu_{2\to 3}(y_2)\mu_{4\to 3}(y_3),\\
\mu_{5\to 3}(y_4) &= f_5(y_4), & \mu_{3\to 2}(y_2) &= \sum_{y_3,y_4} f_3(y_2,y_3,y_4)\mu_{4\to 3}(y_3)\mu_{5\to 3}(y_4),\\
\mu_{3\to 4}(y_3) &= \sum_{y_2,y_4} f_3(y_2,y_3,y_4)\mu_{2\to 3}(y_2)\mu_{5\to 3}(y_4), & \mu_{2\to 1}(y_1) &= \sum_{y_2} f_2(y_1,y_2)\mu_{3\to 2}(y_2),
\end{aligned}$$

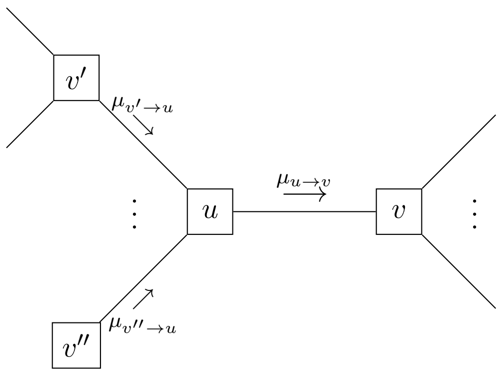


Figure 25: SPA update rule, where the message sent from node u to its neighbor v is computed using all the messages arriving to u from all its other neighbors, and its local function f_u, namely, $\mu_{u\to v} = \langle f_u, \mu_{v'\to u}, \ldots, \mu_{v''\to u}\rangle$.

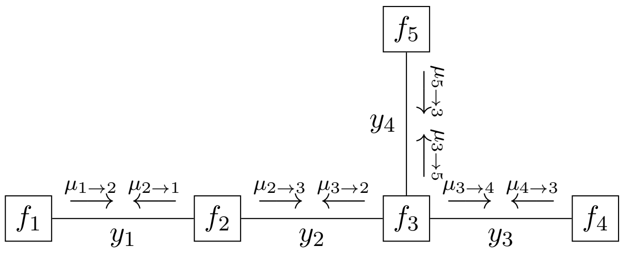


Figure 26: SPA example.

and it is clear by direct substitution that, for instance,

$$Z_{\mathcal{G}}(y_2) = \mu_{2\to 3}(y_2)\mu_{3\to 2}(y_2).$$

Another example is provided below in relation to the difference transform and the "derivative sum-product algorithm" of Huang and Frey [19].

Example 8. *Let $\mathcal{G}$ be as in Fig. 27, the SPA computes the following messages:*

$\mu_{u_1\to d_1}(x_1) = \delta_{\overline{x}_1}(x_1)$;	$\mu_{u_2\to d_2}(x_2) = \delta_{\overline{x}_2}(x_2)$;
$\mu_{d_1\to q_1}(y_1) = \sum_{x_1} D(x_1,y_1)\mu_{u_1\to d_1}(x_1)$;	$\mu_{d_2\to q_2}(y_2) = \sum_{x_2} D(x_2,y_2)\mu_{u_2\to d_2}(x_2)$;
$\mu_{f_1\to q_1}(y_1') = f_1(y_1')$;	$\mu_{q_2\to f_3}(y_2') = \mu_{f_2\to q_2}(y_2')\mu_{d_2\to q_2}(y_2')$;
$\mu_{q_1\to f_2}(y_1'') = \mu_{d_1\to q_1}(y_1'')\mu_{f_1\to q_1}(y_1'')$;	$\mu_{q_2\to f_2}(y_2'') = \mu_{d_2\to q_2}(y_2'')\mu_{f_3\to q_2}(y_2'')$;
$\mu_{f_2\to q_2}(y_2'') = \sum_{y_1''} f_2(y_1'',y_2'')\mu_{q_1\to f_2}(y_1'')$;	$\mu_{f_2\to q_1}(y_1'') = \sum_{y_2''} f_2(y_1'',y_2'')\mu_{q_2\to f_2}(y_2'')$;
$\mu_{f_3\to q_2}(y_2') = f_3(y_2')$;	$\mu_{q_1\to f_1}(y_1') = \mu_{f_2\to q_1}(y_1')\mu_{d_1\to q_1}(y_1')$;
$\mu_{q_2\to d_2}(y_2) = \mu_{f_2\to q_2}(y_2)\mu_{f_3\to q_2}(y_2)$;	$\mu_{q_1\to d_1}(y_1) = \mu_{f_1\to q_1}(y_1)\mu_{f_2\to q_1}(y_1)$;
$\mu_{d_2\to u_2}(x_2) = \sum_{y_2} D(x_2,y_2)\mu_{q_2\to d_2}(y_2)$;	$\mu_{d_1\to u_1}(x_1) = \sum_{y_1} D(x_1,y_1)\mu_{q_1\to d_1}(y_1)$.

By noting that $\mu_{d_1\to q_1}(y_1) = D(\overline{x}_1, y_1)$ and $\mu_{d_2\to q_2}(y_2) = D(\overline{x}_2, y_2)$, one can see by direct substitution that

$$\mu_{d_1\to u_1}(x_1) = \sum_{y_1} D(x_1,y_1)f_1(y_1)\sum_{y_2''} D(\overline{x}_2,y_2'')f_2(y_1,y_2'')f_3(y_2''),$$

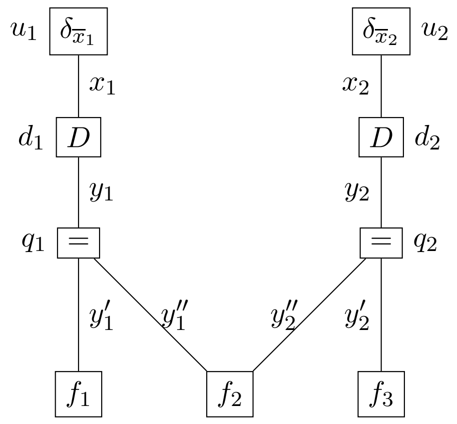


Figure 27:

and

$$\mu_{d_2\to u_2}(x_2) = \sum_{y_2} D(x_2, y_2) f_3(y_2) \sum_{y''_1} D(\overline{x}_1, y''_1) f_2(y''_1, y_2) f_1(y''_1).$$

If we use the notation $\partial_x f(x,y) := \sum_z D(x,z) f(x,y)$ to denote the difference transform of a function f (with respect to x), then the above two messages can equivalently be written as:

$$\mu_{d_1\to u_1}(x_1) = \partial_{x_1}\left[f_1(x_1)\partial_{x_2}[f_2(x_1,x_2)f_3(x_2)]|_{x_2=\overline{x}_2}\right],$$

which is the difference transform of the product of all hidden functions with respect to x_1, x_2 evaluated at $x_2 = \overline{x}_2$, and similarly

$$\mu_{d_2\to u_2}(x_2) = \partial_{x_2}\left[f_3(x_2)\partial_{x_1}[f_2(x_1,x_2)f_1(x_1)]|_{x_1=\overline{x}_1}\right]$$

is the difference transform evaluated at $x_1 = \overline{x}_1$.

Let $\mathcal{G}$ be a tree NFG that is also a constrained NFG with a set I of interface nodes consisting of equality indicators. Let $\mathcal{G}'$ be the NFG obtained from $\mathcal{G}$ by gluing on each dangling edge a difference function, and $\mathcal{G}''$ be the NFG resulting from $\mathcal{G}'$ by gluing an evaluation function on the other end of each such difference function, cf. Fig. 27. Performing the SPA on $\mathcal{G}''$, it is clear that the discussion in the previous example extends to any such $\mathcal{G}''$. That is, the message $\mu_{d_i\to u_i}(x_i)$ is the difference transform of the product of the hidden functions with respect to x_I evaluated at $\overline{x}_{I\setminus\{i\}}$, where d_i is the difference node adjacent to interface node i and u_i is the evaluation node adjacent to d_i. This is the "derivative-sum-product" algorithm of [19] in the case of finite alphabets.

We close with two remarks: First, we emphasize that in performing the SPA over $\mathcal{G}''$, one may utilize the fast difference algorithm in computing the messages emitted by the difference nodes, making the complexity of computing such messages linear in the alphabet size. Second, as one may expect, marginalization by summation on $\mathcal{G}'$ is marginalization by evaluation on $\mathcal{G}$, and the difference function guarantees the conversion between such notions of marginalization, as demonstrated in the example below.

Example 8 (Continued). *Assume the function of node u_1 in the NFG in Fig. 27 is replaced with the constant-one indicator (this accounts to marginalizing x_1 by summing it out), and assume all* *variables take their values from a set $\mathcal{X}$. Then we have $\mu_{u_1\to d_1}(x_1)=1$, and from the definition of the difference function, it is clear that $\mu_{d_1\to q_1}(y_1)=\delta_{|\mathcal{X}|}(y_1)$. By direct substitution, it follows that*

$$\mu_{d_2\to u_2}(x_2)=\sum_{y_2} D(x_2,y_2)f_1(|\mathcal{X}|)f_2(|\mathcal{X}|,y_2)f_3(y_2).$$

Or equivalently,

$$\mu_{d_2\to u_2}(x_2)=\partial_{x_2}\big[f_1(|\mathcal{X}|)f_2(|\mathcal{X}|,x_2)f_3(x_2)\big]$$

That is, the message $\mu_{d_2\to u_2}$ is the difference transform with respect to x_2 of the multiplication of the hidden functions, with x_1 marginalized by evaluation at $|\mathcal{X}|$. One may verify that $\mu_{d_1\to u_1}$ remains unchanged.

7.3 Indirect evaluation of the exterior function

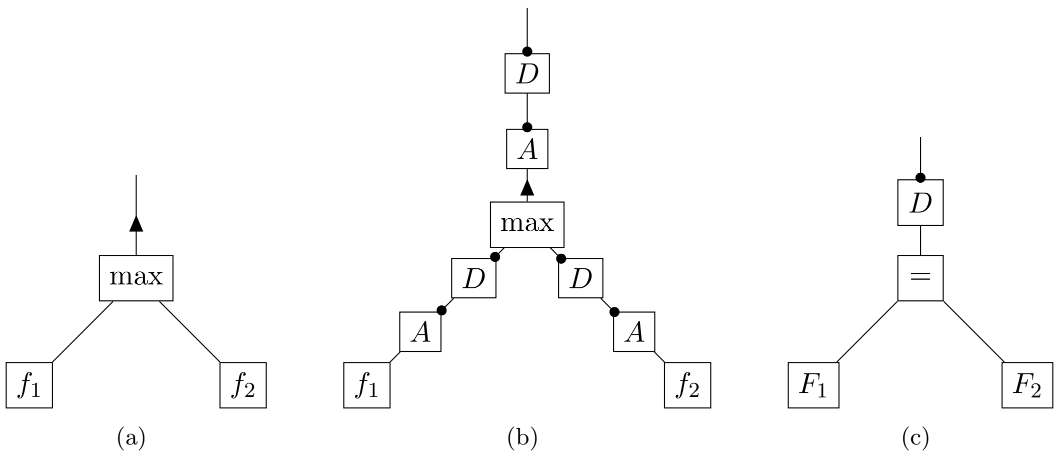
Below, we discuss an indirect method for finding the exterior function of a bipartite NFG $\mathcal{G}$, where a holographic transformation, a one that preserves the exterior function, is applied to the NFG *a priori* in hope of facilitating the computation. The idea is to perform a transformation, if it exists, that replaces each function f_i with an equality indicator, and hence benefit from the low computational complexity of such nodes in the elimination or the SPA. Of course one might not be able to find a holographic transformation that converts each function f_i into an equality indicator, in which case, one may settle with one that produces such effect for a subset $I'\subseteq I$. For this approach to work, it is necessary that: 1) There exists a set of transformers $\{g_e : e\in E\}$ such that $\big\langle f_i, g_e : e\in E(i)\big\rangle=\delta_=$, for all $i\in I'$ for some non-empty I', and 2) There exists efficient algorithms for computing the transformations induced by g_e. The following example demonstrates this approach.

Example 9. *Let $\mathcal{X}$ be a finite ordered set and consider the NFG in Fig. 28 (a) where each edge variable assumes its values from the partially-ordered set $\mathcal{X}^n$ for some integer n. Directly computing the exterior function requires $|\mathcal{X}|^{2n}$ operations. However, the exterior function may be computed using a number of operations of order $n|\mathcal{X}|^n$ by first computing the fast cumulus transforms of f_1 and f_2, multiply the resulting two functions, and then invert the result using the fast difference transform, Fig. (c). The exterior function is invariant under such procedure since it simply accounts to the holographic transformation in Fig. (b), which is equivalent to the NFG in (a) by the GHT and to the one in (c) by Lemma 5.*

The discussion above parallels the well-known fast Fourier transform approach to finding the convolution of functions. In this case, the fast Fourier transform is used to reduce the complexity of computing the convolution of two functions defined on $\mathcal{X}^n$ (assuming $\mathcal{X}$ is a finite abelian group) from order $|\mathcal{X}|^{2n}$ to $n\log(|\mathcal{X}|)|\mathcal{X}|^n$ by first computing the fast Fourier transforms of f_1 and f_2, multiply the resulting two functions, and then invert the result using the fast Fourier inverse transform. This is justified by the relation between the sum and parity indicators, and the duality of the equality and parity indicators under the Fourier transform, Figs 28 (d)–(f).

8 The inference problem

Given an NFG $\mathcal{G}(V,E,f_V)$ representing a set of RVs X_L. (That is, the exterior function of $\mathcal{G}$ is the probability distribution p_{X_L}.) The inference problem is to marginalize a set $M\subseteq L$ of the RVs and



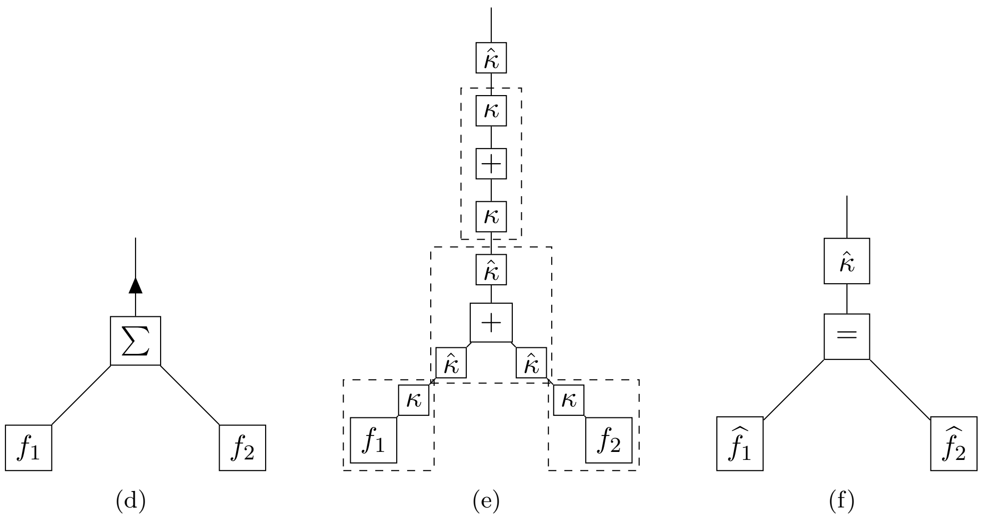


Figure 28: Indirect computation of the exterior function, where F_j, and $\widehat{f_j}$ are the cumulus and Fourier transforms of f_j, respectively.

to evaluate p_{X_L} at some observed values (evidence) of RVs $N \subseteq L$, where M and N are disjoint. That is, the problem is to find

$$p_R(x_R, \overline{x}_N) = \sum_{x_M} p_{X_L}(x_R, x_M, x_N)|_{x_N = \overline{x}_N},$$

where $R = L \backslash (M \cup N)$. We remark that, in general, p_R is not a probability distribution over the RVs X_R. It is rather an *up to scale* distribution over X_R, namely, it is the conditional distribution $p_{X_R|x_N}(x_R|x_N = \overline{x}_N)$ up to the scaling constant $p_{X_N}(\overline{x}_N)$. Evidently, the complexity of the inference problem depends primarily on the factorization structure of p_{X_L}, which is reflected by the graphical structure of the NFG. In order to perform the desired inference, we define $\mathcal{G}^*$ as the NFG whose exterior function is the desired function p_R. That is, we define $\mathcal{G}^*$ as the NFG obtained from $\mathcal{G}$ by: 1) Converting each dangling edge $e \in M$ into a regular edge by gluing a new vertex u_e to e, where u_e is associated the constant-one function. 2) Convert each dangling edge $e \in N$ into a regular edge by gluing a new vertex v_e to e, where v_e is associated the evaluation indicator $\delta_{\overline{x}_e}$. An example is shown in Fig. 29 where the original NFG is as in (a). Assuming we are interested in $\sum_{x_3} p_{X_1X_2X_3}(x_1, x_2, x_3)|_{x_2=\overline{x}_2}$, then $\mathcal{G}^*$ is as in (b).

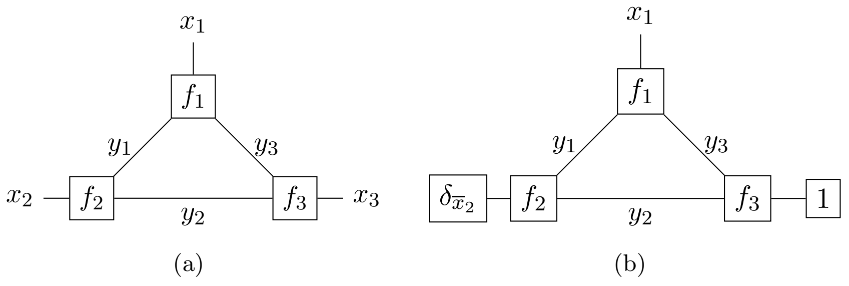


Figure 29: Inference: (a) An example NFG $\mathcal{G}$ representing $p_{X_1X_2X_3}(x_1, x_2, x_3)$, (b) the resulting $\mathcal{G}^*$ assuming we are interested in $\sum_{x_3} p_{X_1X_2X_3}(x_1, x_2, x_3)|_{x_2=\overline{x}_2}$.

Clearly the inference problem is encoded in $\mathcal{G}^*$, and hence reduces to computing the exterior function of $\mathcal{G}^*$, which can be performed by invoking the elimination algorithm on $\mathcal{G}^*$. From this equivalence between inference and the computation of the exterior function, one can always assume that the given NFG represents the desired computation, i.e., one can assume that the NFG is already reduced to the desired inference $(.)^*$ form.

We remark that in a constrained model, by Proposition 2, we may assume that each interface node is an equality indicator. Hence, for each evaluated RV, i.e., for each $i \in N \subseteq I$, we may 1) for each neighbor $j \in \text{ne}(i)$ of i, connected to i by edge e, replace f_j with $f_j(x_{E(j)})|_{x_e=\overline{x}_e}$ and delete e, and 2) delete node i. Hence, the inference problem over constrained models is simply a marginalization one, and one may always assume N is empty.

On the other hand, for a conditional function of x given y, we have $\sum_x f(x, y)$ is a constant c independent of y. Hence, in a generative model $\mathcal{G}$ with vertex set $I \cup J$, for each marginalized RV, i.e., for each $i \in M \subseteq I$, we may 1) absorb the constant c_i into one of the neighbors of i by replacing f_j with $c_i f_j$ for some $j \in \text{ne}(i)$, 2) for each neighbor $j \in \text{ne}(i)$ of i, connected to i by edge e, replace f_j with $\sum_{x_e} f_j(x_{E(j)})$ and delete e, and 3) delete node i. Hence, the inference problem over generative models is simply an evaluation one, and one may always assume M is empty.

9 Concluding Remarks

In this paper we presented NFGs as a new class of probabilistic graphical models. We showed that this framework and the transformation technique herein unify various previous models, including multiplicative models, convolutional factor graphs, and the known transform-domain models. We focused on two dual categories of NFG models, constrained and generative NFG models, and revealed an interesting duality in their implied dependence structure.

We feel that approaching learning and inference problems in a transform domain is methodologically appealing. The generic and flexible transformation technique introduced in this framework may potentially demonstrate great power along that direction.

It is our hope that the idea and modelling framework presented in this paper find new applications beyond the reach of conventional models.

References

[1] J. Pearl, *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, 1988.

[2] R. Kindermann and J. L. Snell, *Markov random fields and their applications.* American Mathematical Society Providence, 1980.

[3] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[4] B. J. Frey, "Extending factor graphs so as to unify directed and undirected graphical models," in *Proceedings of the 19th conference on Uncertainty in Artificial Intelligence*, 2002, pp. 257–264.

[5] Y. Mao, F. R. Kschischang, and B. J. Frey, "Convolutional factor graphs as probabilistic models," in *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, 374–381.

[6] Y. Mao and F. R. Kschischang, "On factor graphs and the Fourier transform," *IEEE Trans. Inform. Theory*, vol. 51, no. 5, pp. 1635–1649, 2005.

[7] D. Bickson and C. Guestrin, "Inference with multivariate heavy-tails in linear models," in *Neural Information Processing System (NIPS)*, Vancouver, Canada, Dec. 2010.

[8] J. C. Huang and B. J. Frey, "Cumulative distribution networks and the derivative-sum-product algorithm," in *Proceedings of the 24th conference on Uncertainty in Artificial Intelligence*, 2008.

[9] A. Al-Bashabsheh and Y. Mao, "Normal factor graphs and holographic transformations," *IEEE Trans. Inform. Theory*, vol. 57, no. 2, pp. 752–763, Feb. 2011.

[10] G. D. Forney, Jr., "Codes on graphs: Duality and MacWilliams identities," *IEEE Trans. Inform. Theory*, vol. 57, no. 3, pp. 1382–1397, 2011.

[11] ——, "Codes on graphs: Normal realizations," *IEEE Trans. Inform. Theory*, vol. 51, no. 2, pp. 520–548, 2001.

[12] L. G. Valiant, "Holographic algorithms (extended abstract)," in *Proc. 45th Annual IEEE Symp. on Foundations of Computer Science*, Rome, Italy, Oct. 2004, pp. 306–315.

[13] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Sig. Proc. Mag.*, vol. 21, no. 1, pp. 24–41, 2004.

[14] H.-A. Loeliger and P. O. Vontobel, "A factor-graph representation of probabilities in quantum mechanics," in *IEEE Int. Symp. Information Theory*, Cambridge, USA, July 2012.

[15] S. L. Lauritzen, *Graphical Models.* Oxford University Press, 1996.

[16] G. D. Forney, Jr., Principles of digital communication II. Printed notes for MIT course 6.451, 2005.

[17] M. I. Jordan, *An introduction to probabilistic graphical models*, 2002, unpublished draft.

[18] G. D. Forney, Jr. and P. O. Vontobel, “Partition functions of normal factor graphs,” in *Proc. Information Theory and Applications Workshop*, San Diego, CA, Feb. 2011.

[19] J. C. Huang and B. J. Frey, “Cumulative distribution networks and the derivative-sum-product algorithm: Models and inference for cumulative distribution functions on graphs,” *The Journal of Machine Learning Research*, vol. 12, pp. 301–348, 2011.